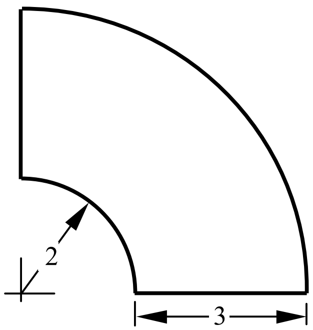
Geometry Modeling
&
Grid Generation

Geometry Modeling & Grid Generation

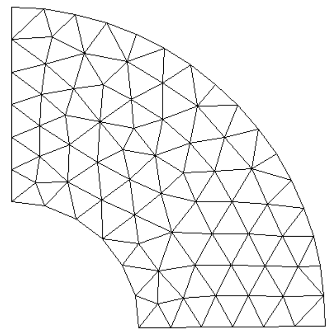
- Geometry definition (simple shapes, CAD import)
- Grid generation algorithms
- GAMBIT
- Grid quality and improvement
- Automation

Acknowledgements:
Fluent Inc. Gambit User Manual
S. Owen: Introduction to unstructured mesh generation

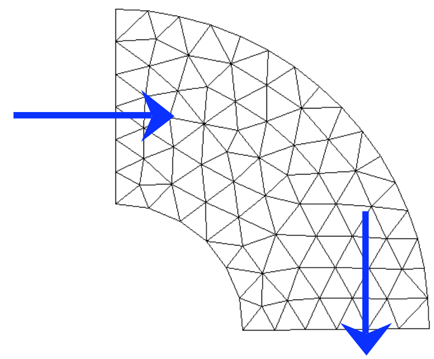
Simulation Process



1. Build CAD Model



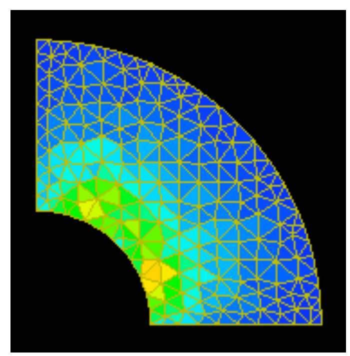
2. Mesh



3. Apply Boundary Conditions

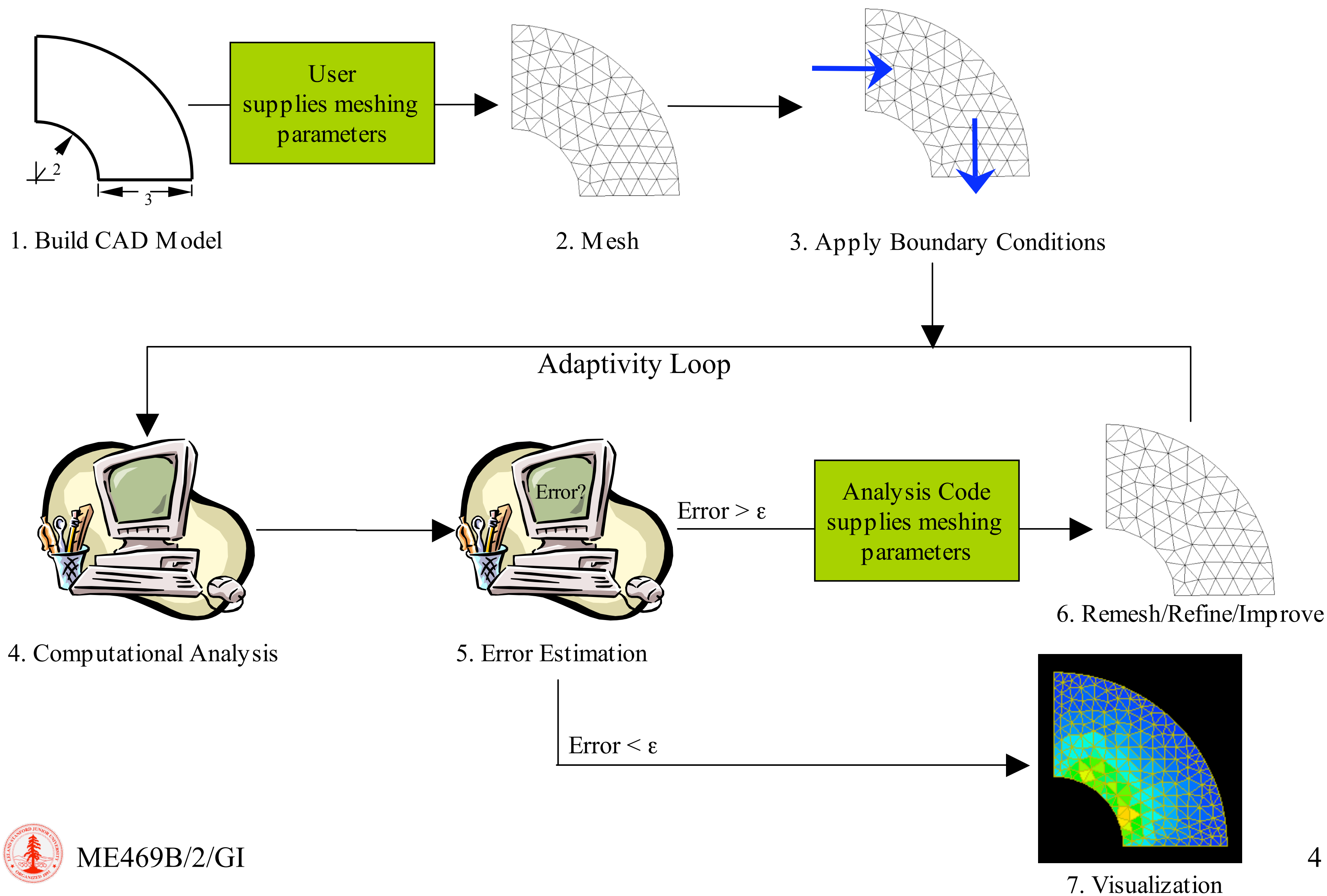


4. Computational Analysis

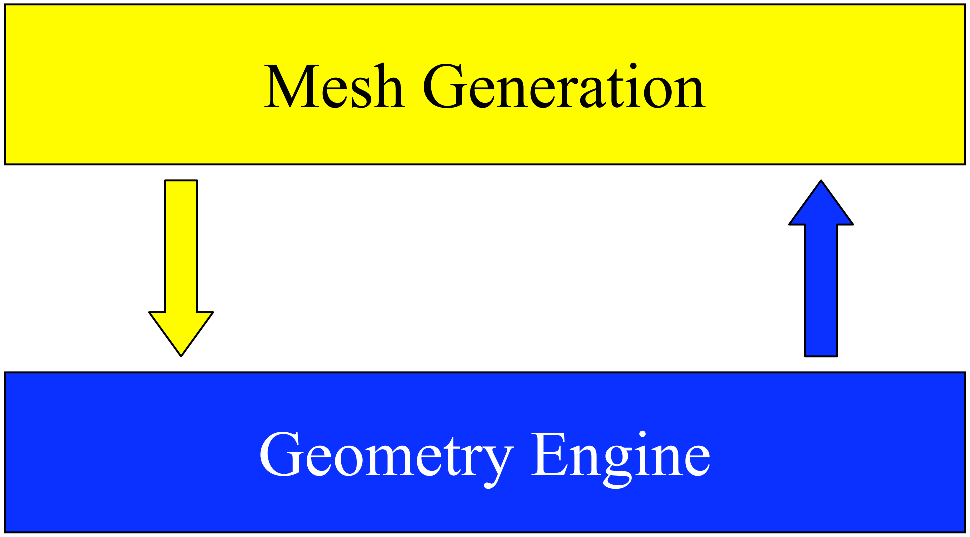


5. Visualization

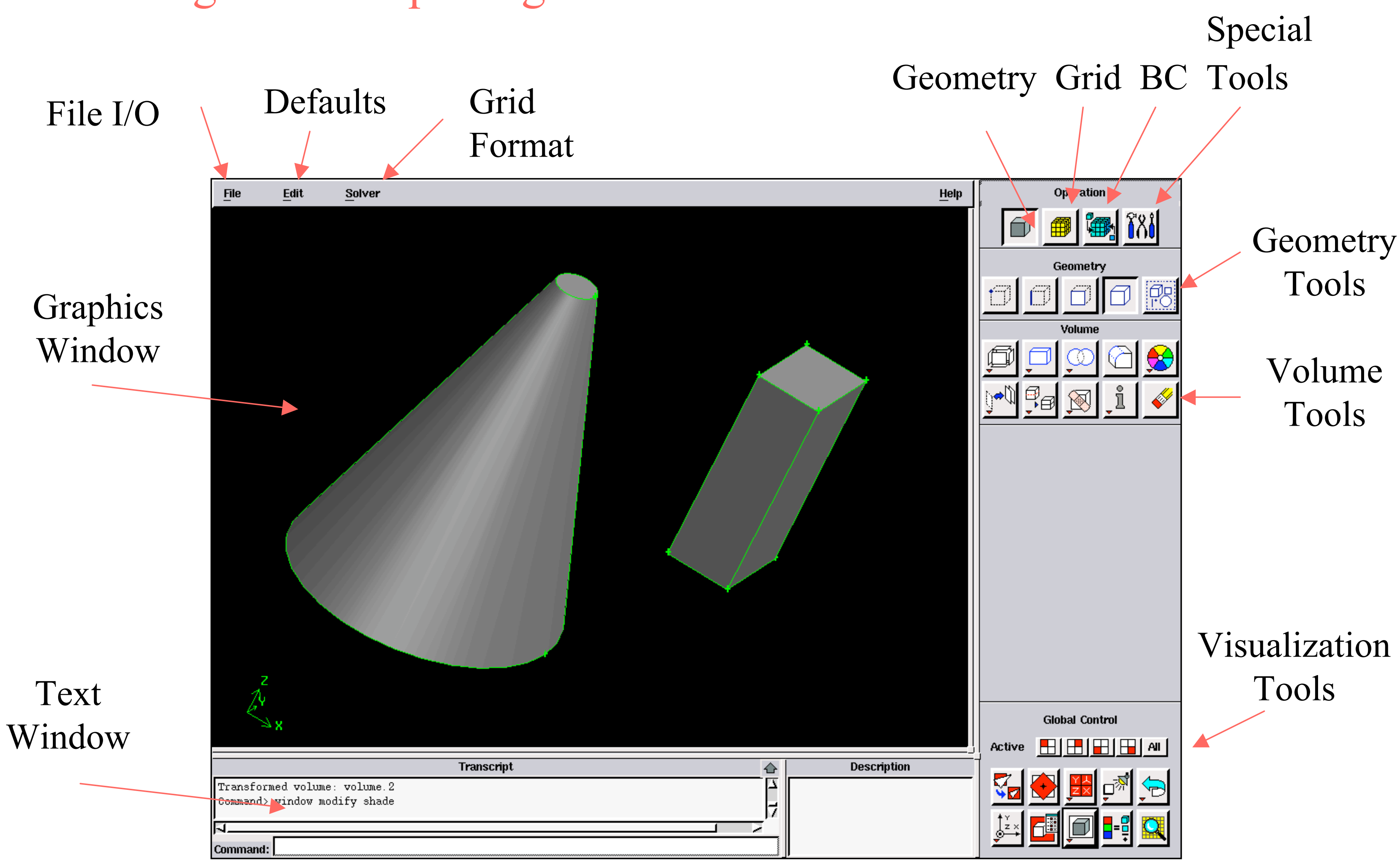
Adaptive Simulation Process



Geometry



Grid generation package: GAMBIT



GAMBIT

`gambit -id <namefile>`

Interactive execution with GUI

`gambit -inp <journalfile>`

Batch execution without GUI

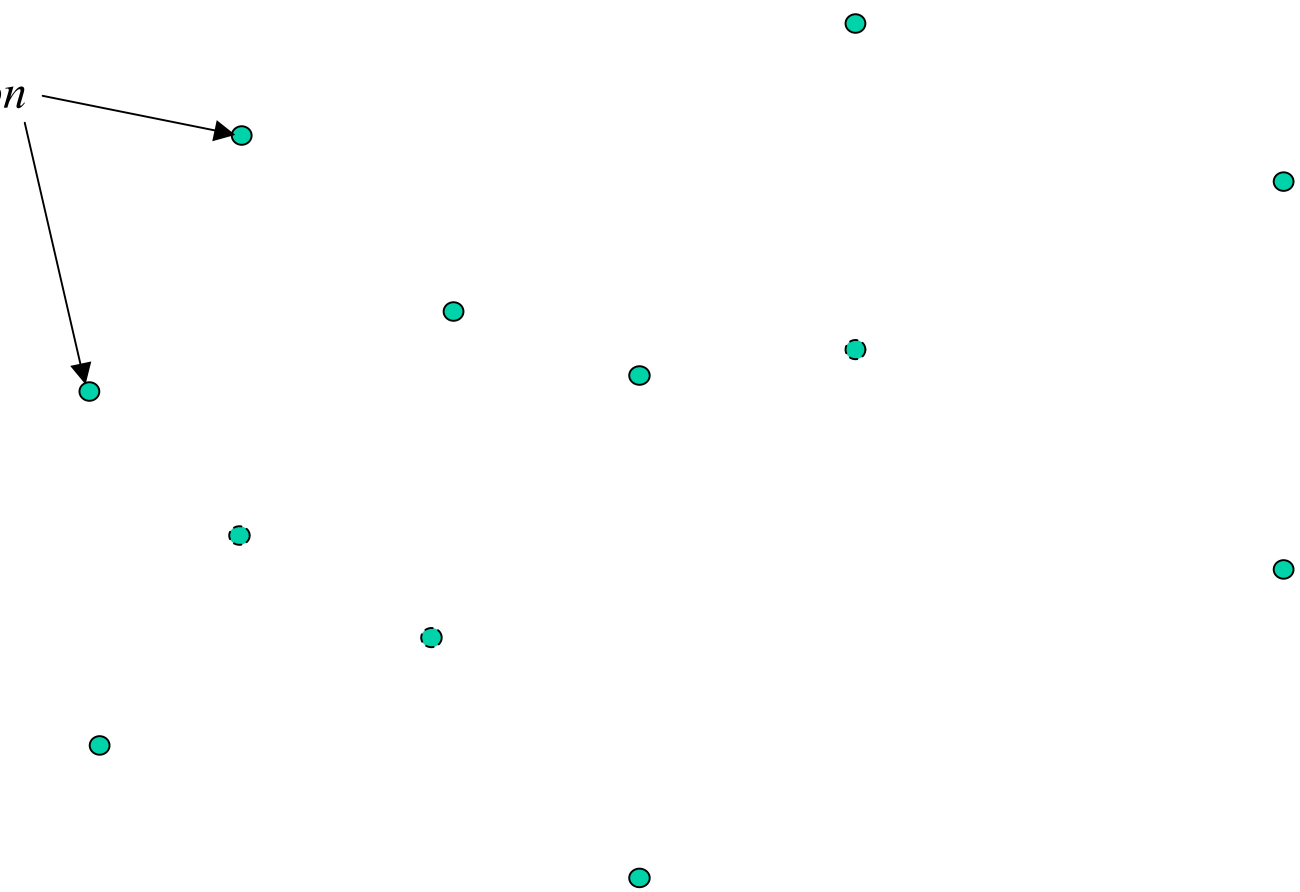
Batch and GUI execution are EXACTLY equivalent!

Geometry & Grid are saved in a *database* file (*.dbs)
The mesh is saved into a solver-dependent file (*.msh)

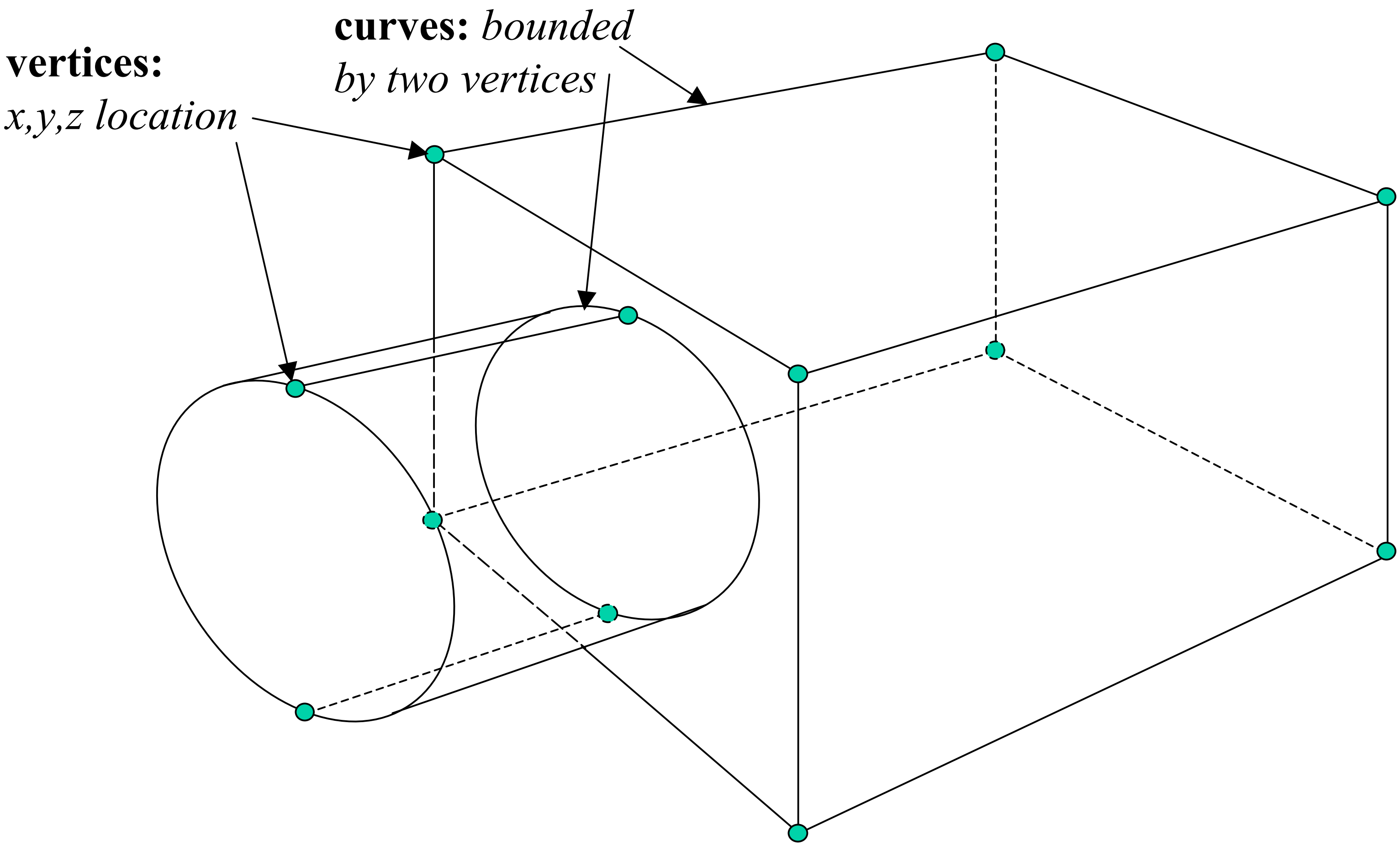
At the end of each session Gambit automatically saves a journal file (*.jou)

Geometry

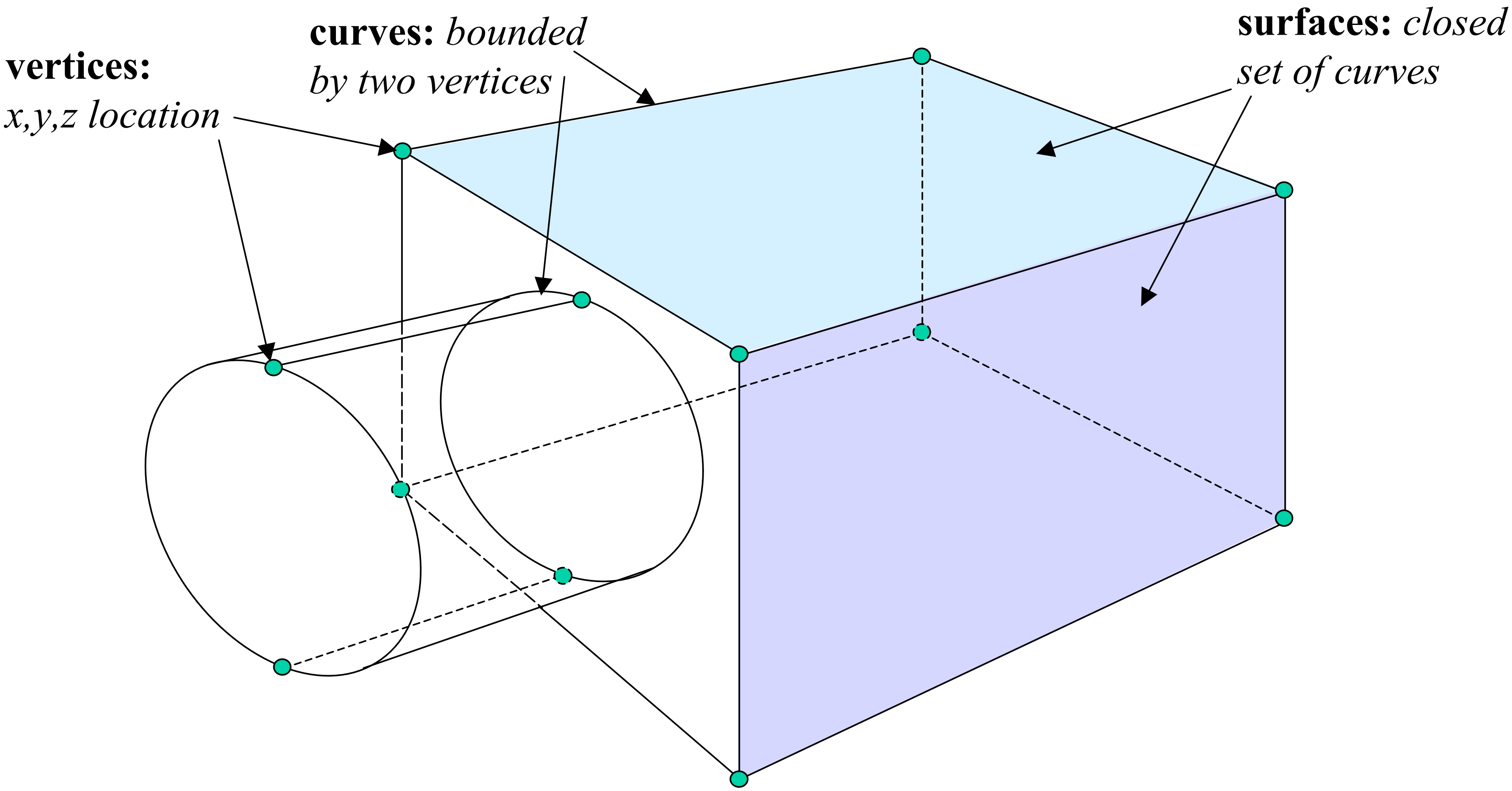
vertices:
x,y,z location



Geometry

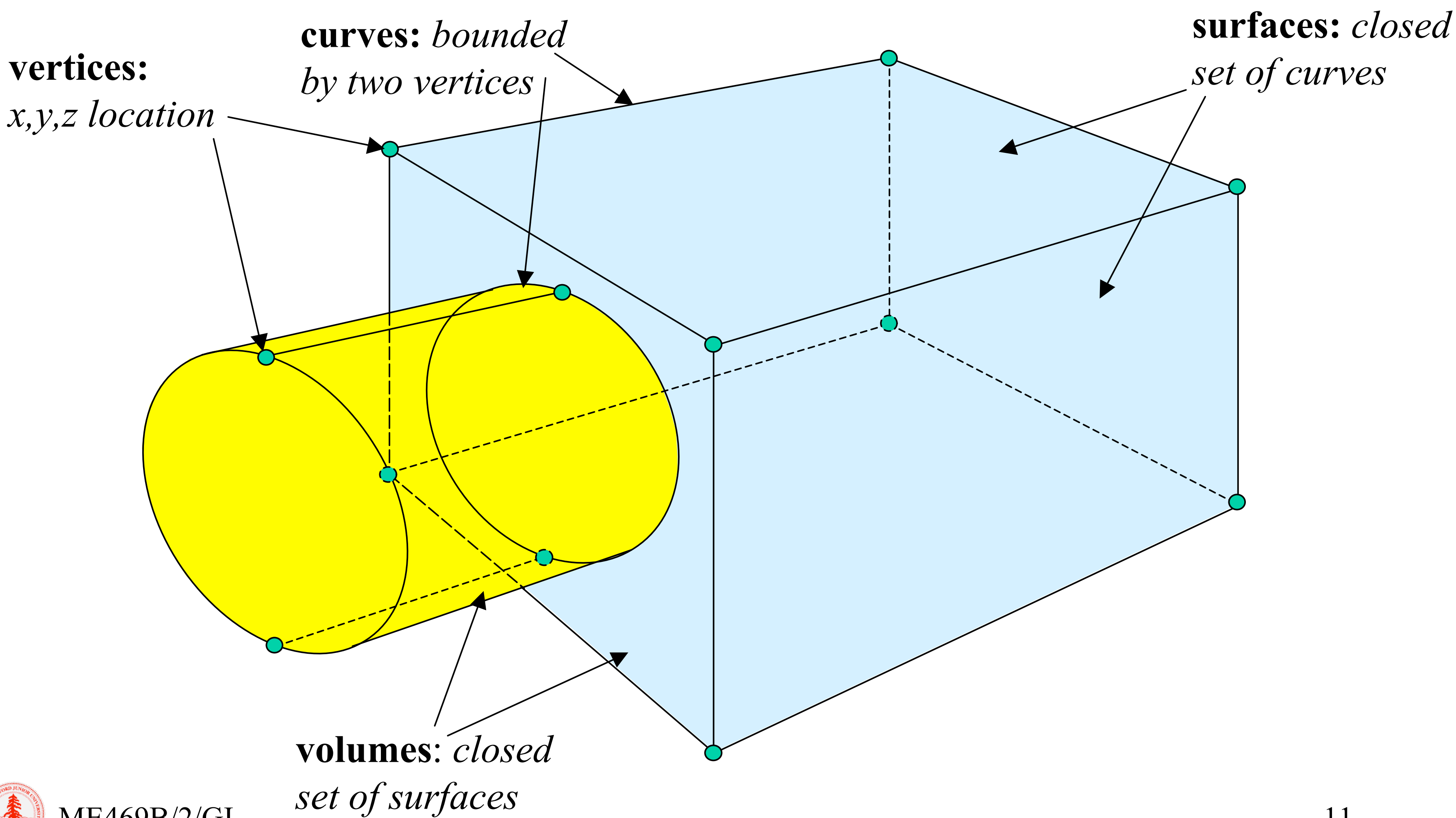


Geometry

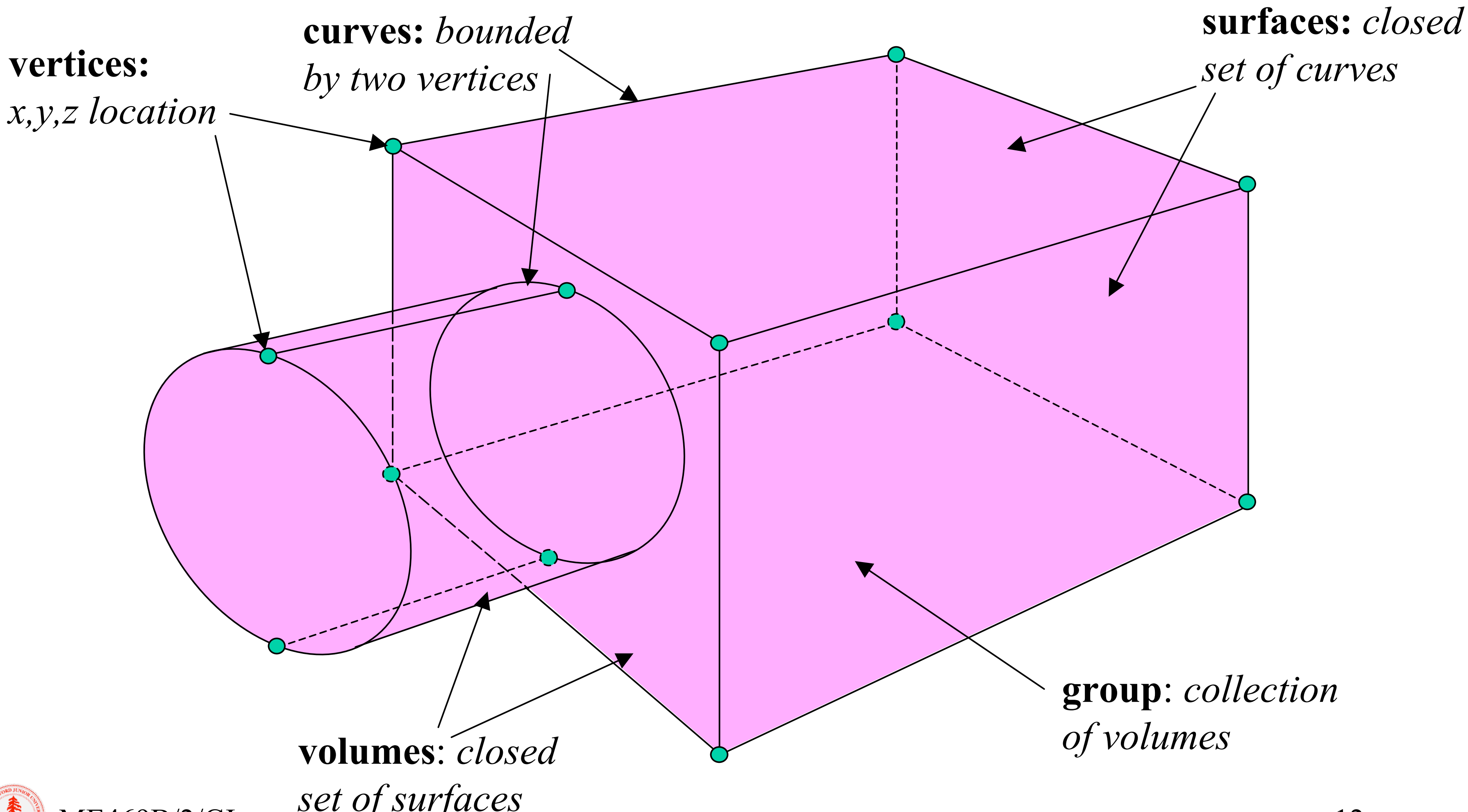


vertices: *x,y,z location*
curves: *bounded by two vertices*
surfaces: *closed set of curves*

Geometry

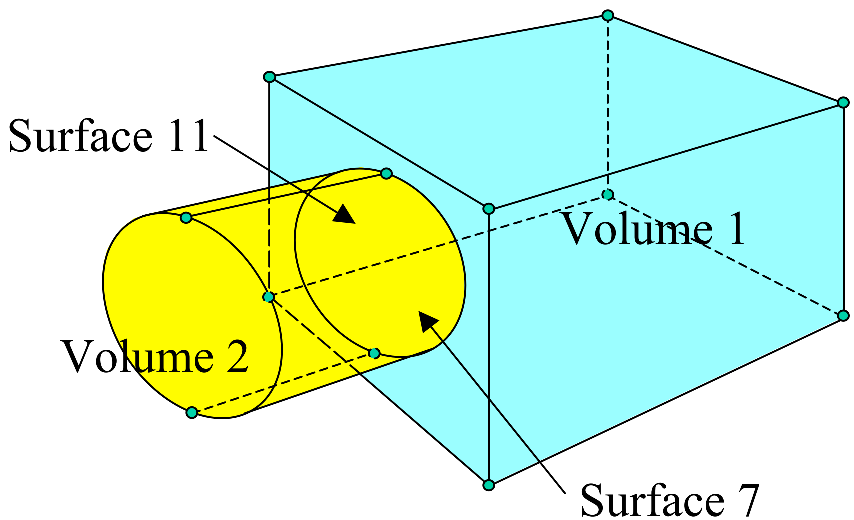

vertices:
x,y,z location
curves: *bounded*
by two vertices
surfaces: *closed*
set of curves
volumes: *closed*
set of surfaces

Geometry

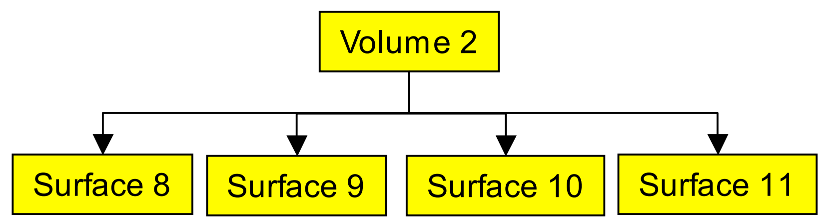


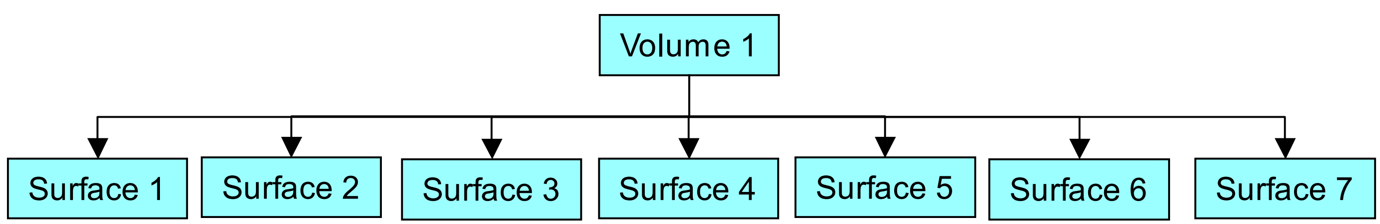
vertices:
x,y,z location
curves: *bounded by two vertices*
surfaces: *closed set of curves*
volumes: *closed set of surfaces*
group: *collection of volumes*

Geometry

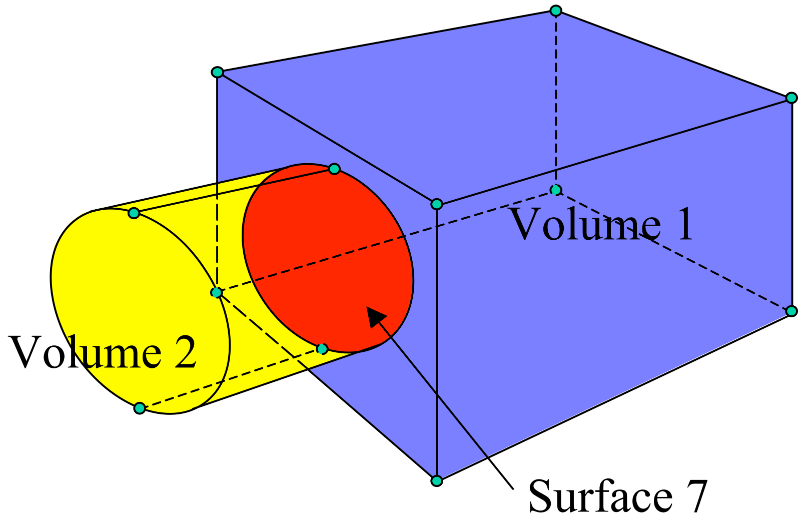


Manifold Geometry:
Each volume maintains its own set of unique surfaces

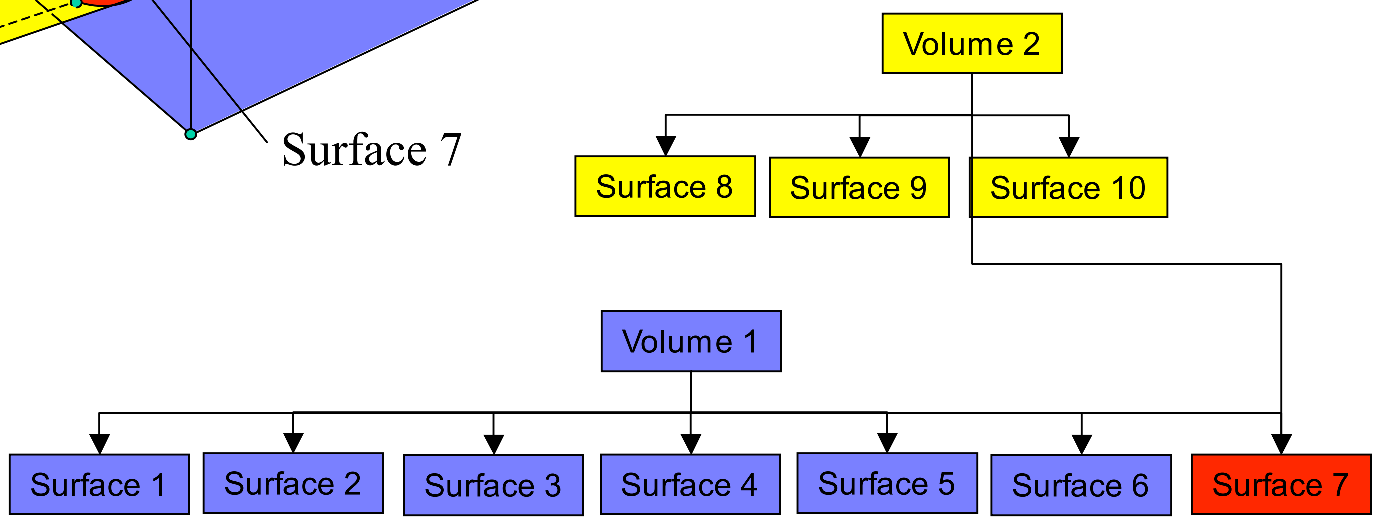




Geometry



Non-Manifold Geometry: *Volumes share matching surfaces*



Geometry & Topology

Geometry types in Gambit

- Real Geometry: entities characterized by a direct definition of their geometry

  example: a vertex defined by its coordinates (0,0,0)

- Virtual Geometry: entities characterized ONLY by an indirect definition, i.e. a reference to another entity.

  example: a vertex is defined as the mid-point of an edge

- Faceted Geometry : entities characterized ONLY by an indirect definition with respect to an underlying grid

  example: a vertex is defined as the corner of a mesh element

Geometrical types - Topology

- Vertex
- Edge (2 or more vertices)
- Face (3 or more edges)
- Volume (4 or more faces)

Bottom-up approach: generate low dimensional entities and build on top of them higher dimensional entities

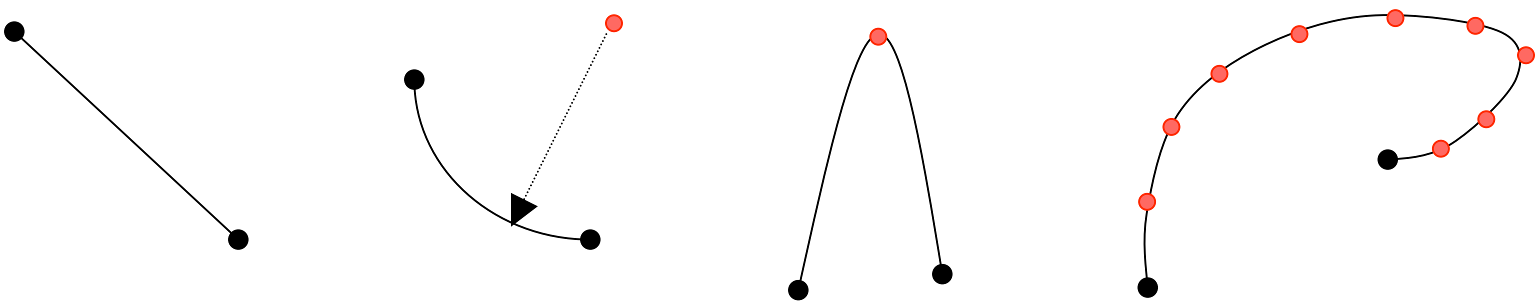
Top-bottom approach: generate upper dimensional entities and use boolean operation to define the other entities

Vertex:

- Input Coordinates…

Edge:

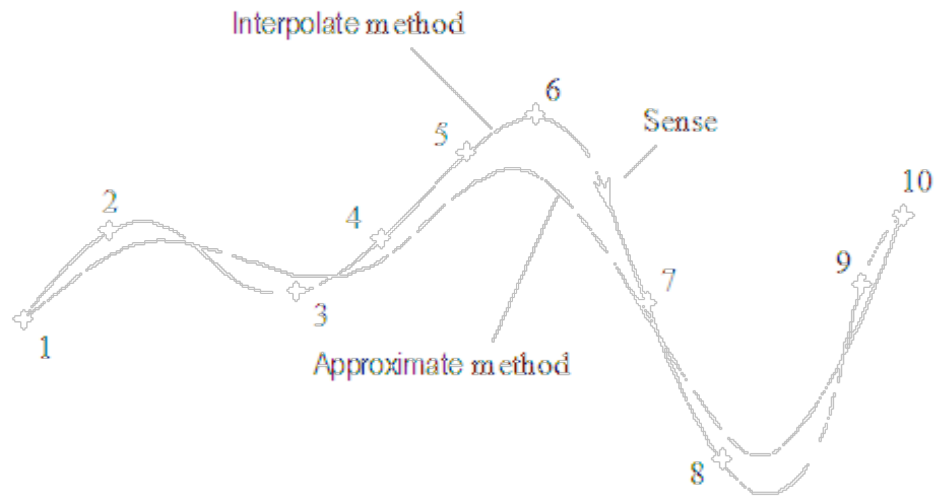
- Line segment (connect 2 vertices)
- Circular arc
- Quadratic functions
- NURBS: Non-Uniform Rational B-Splines (connect N vertices)



Topologically any edge is ALWAYS a connection between 2 vertices
(additional vertices used to build the geometry are NOT part of the edge)

Edge by NURBS

Non-Uniform Rational B-Splines



Generalization of Bezier interpolants: each point is computed as the weighted sum of all the knot points

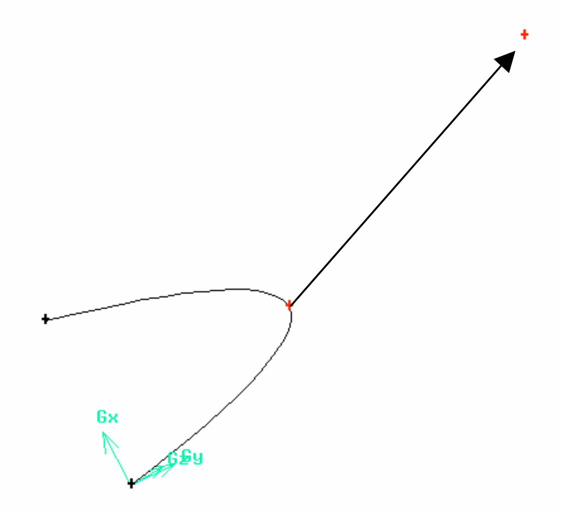
NURBS can use various blending/control functions (for the weights)

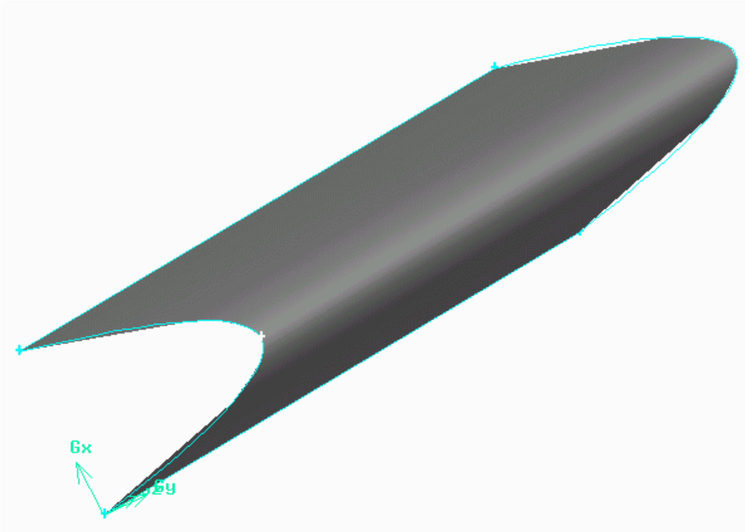
Can achieve high degree of continuity

http://www.ibiblio.org/e-notes/Splines/NURBS.htm

Face:

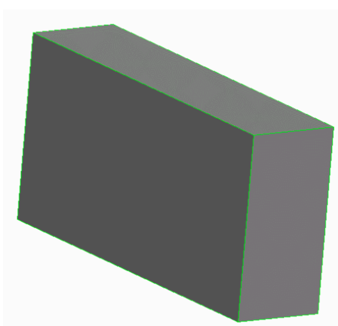
- Rectangular
- Circular
- …
- Sweep (translation or rotation of an edge)
- Wireframe (connecting 3 or more edges)

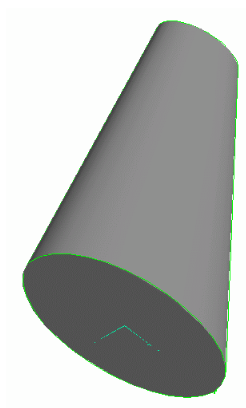


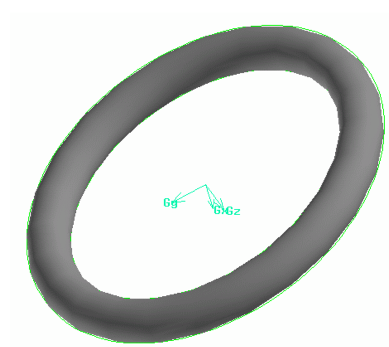


Volume:

- Cuboid
- Sphere
- Cone
- Pyramids
- …
- Sweep (translation or rotation of a face)
- Wireframe (connecting 3 or more faces)





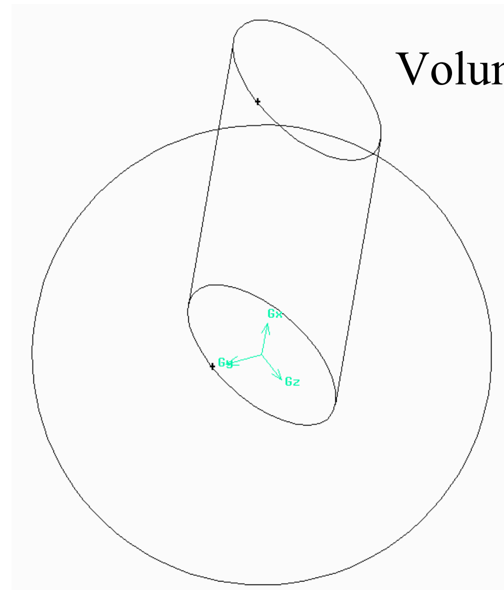


Manipulate Geometry - Boolean Operations:

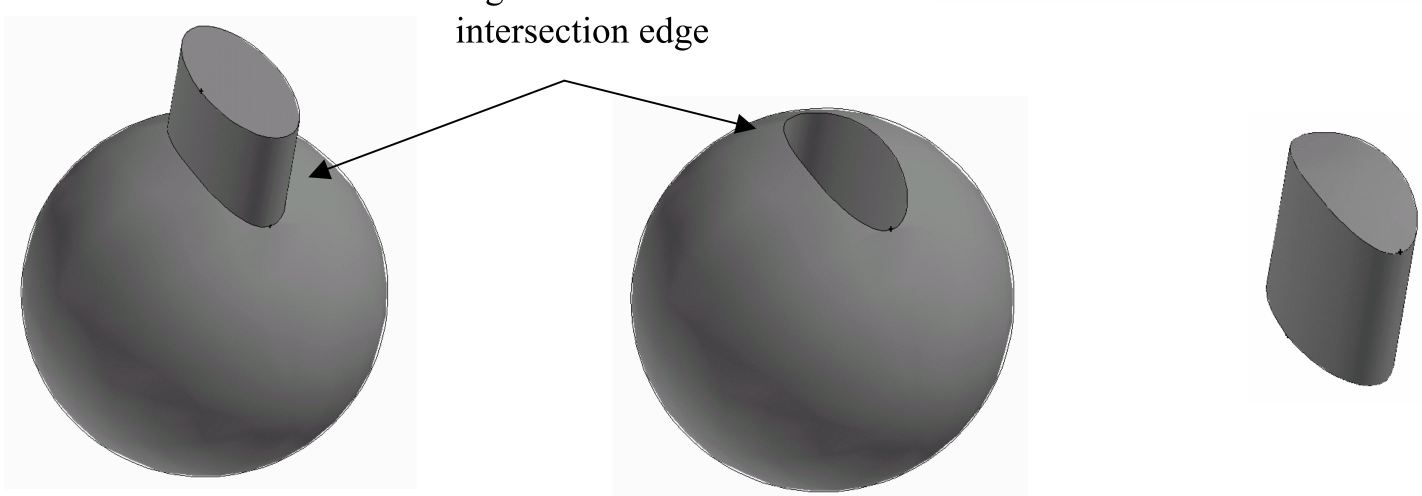
- Unite
- Substract
- Intersect

Volume 1

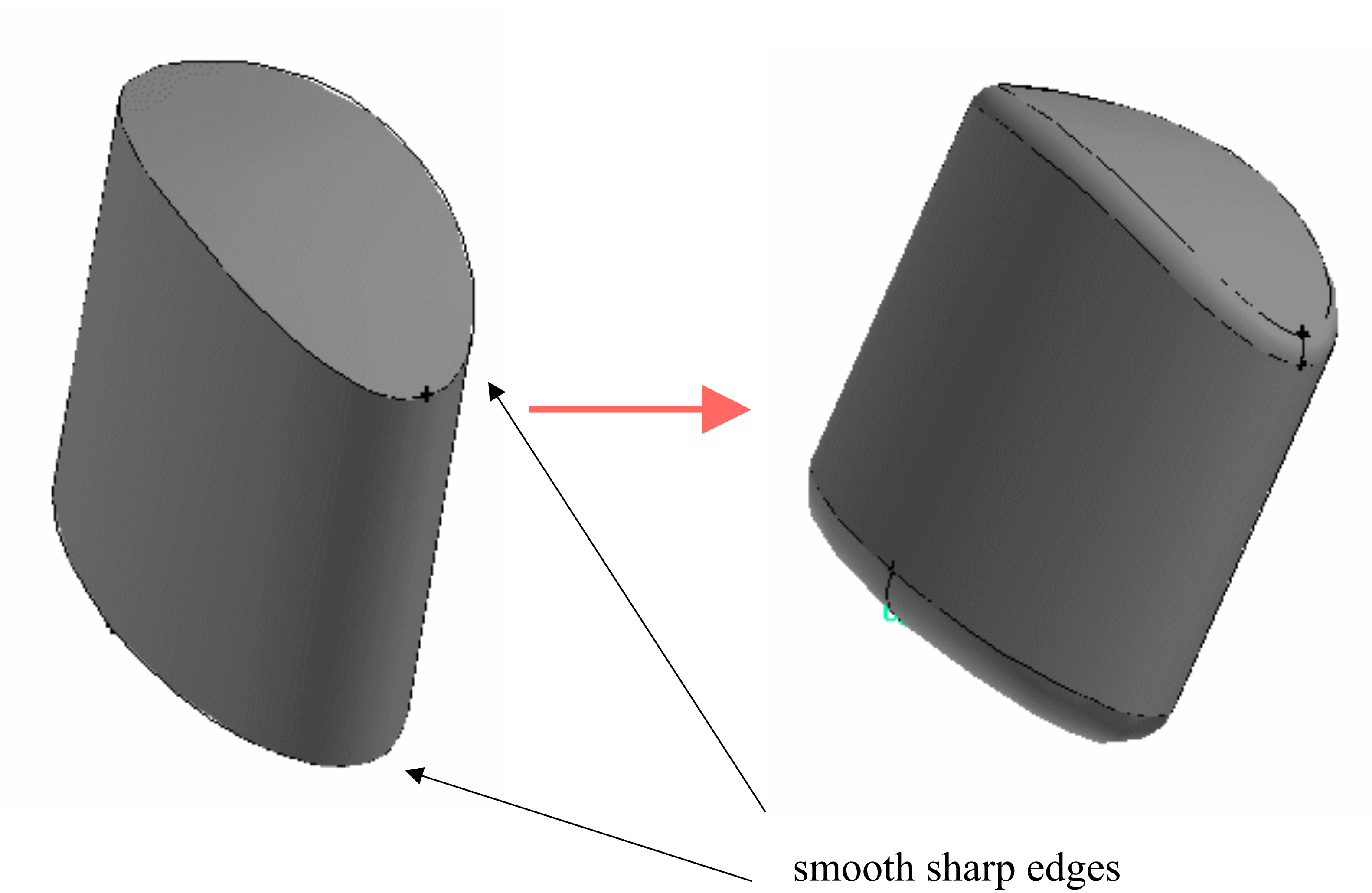
Volume 2



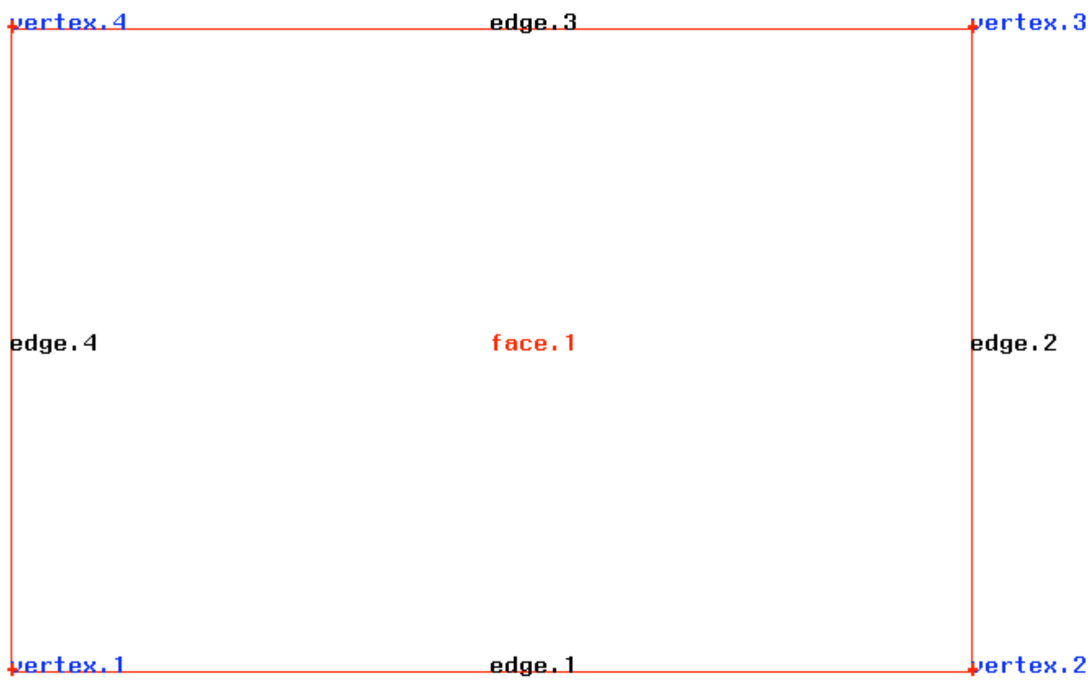
it generates the
intersection edge

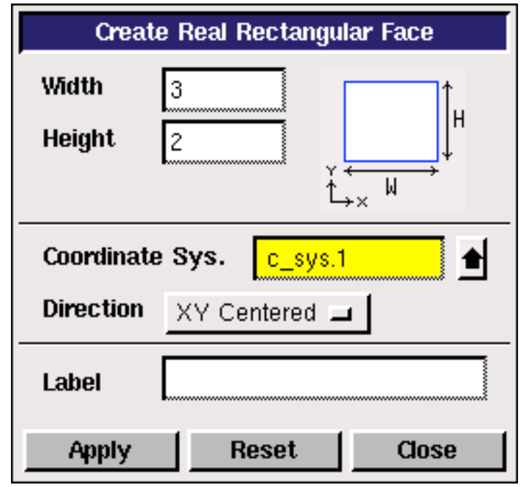


Manipulate Geometry - Blend

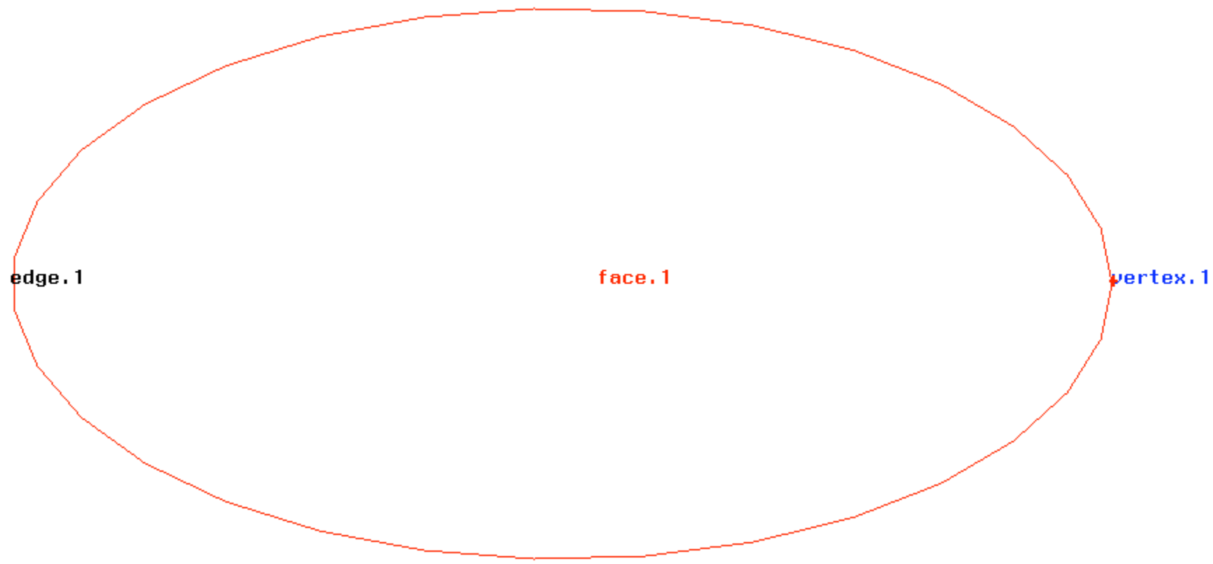


Create Entities - Faces





Create Entities - Faces



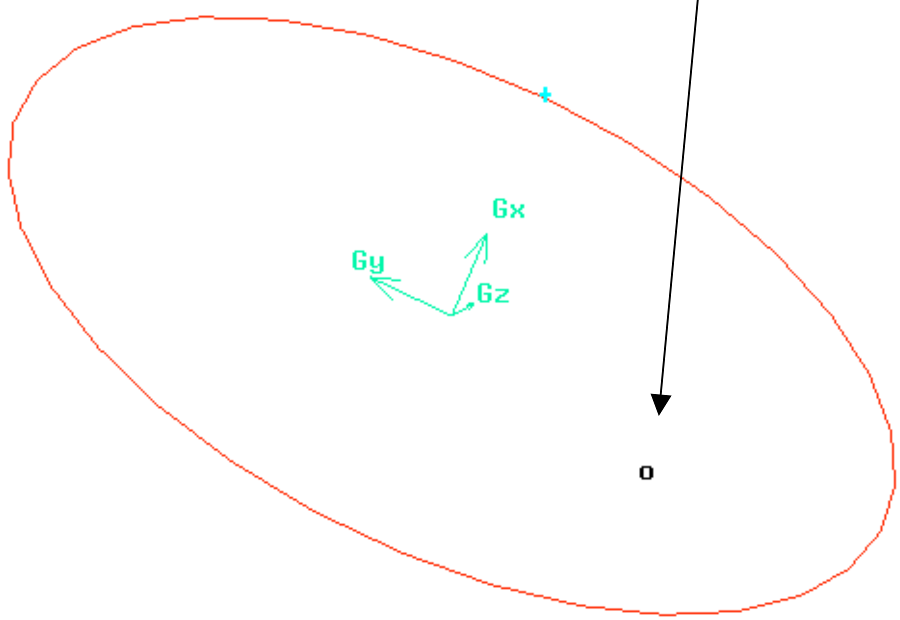
Some entities generated using "primitives" have fewer lower topological entities
Example:

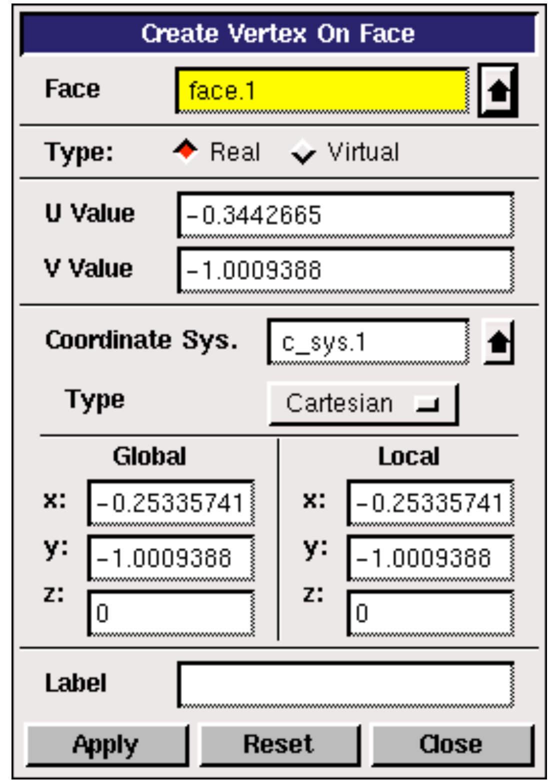
Cube volume: 6 faces, 12 edges, 8 vertices
Cylinder volume: 3 faces, 2 edges, 2 vertices

Manipulate Geometry – Create Entities

Create a vertex on a face

Parametric representation of the face

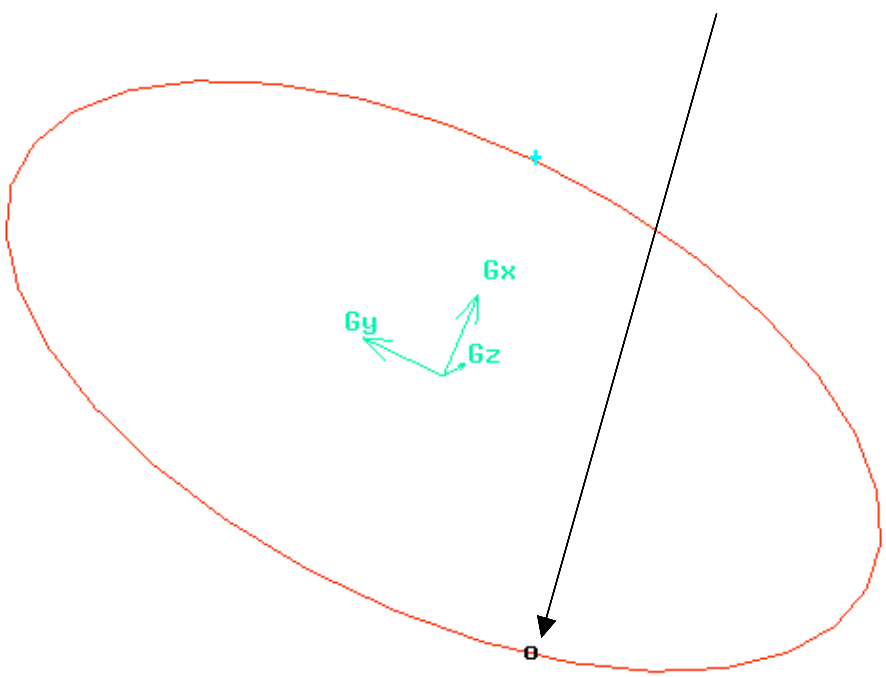


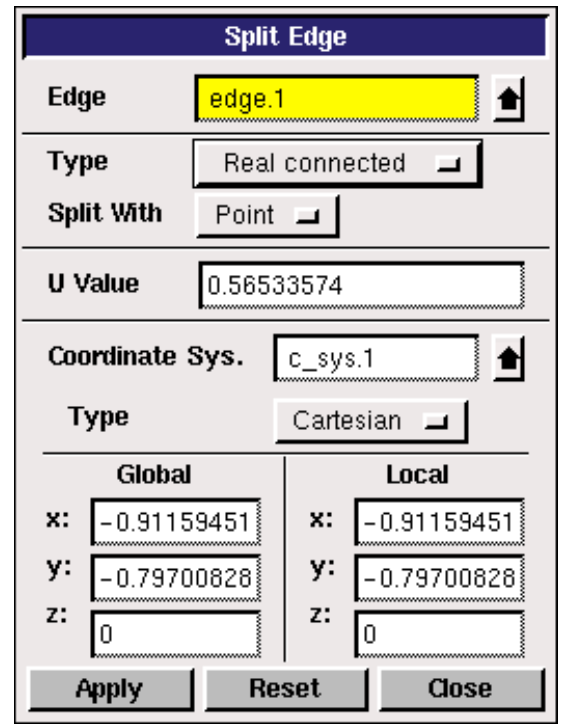


Manipulate Geometry – Create Entities

Create two edges by splitting an edge

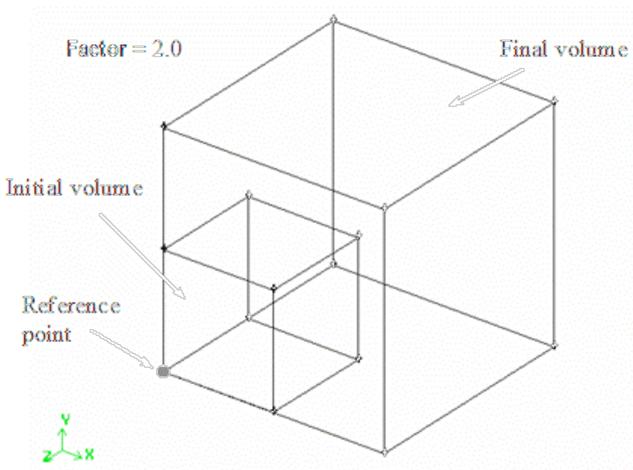
Parametric representation of the edge

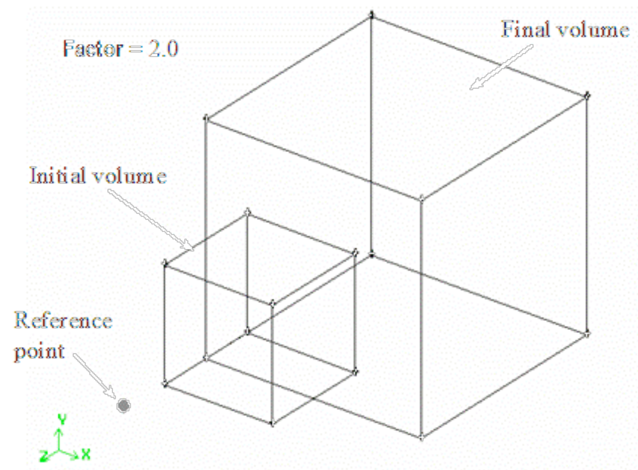


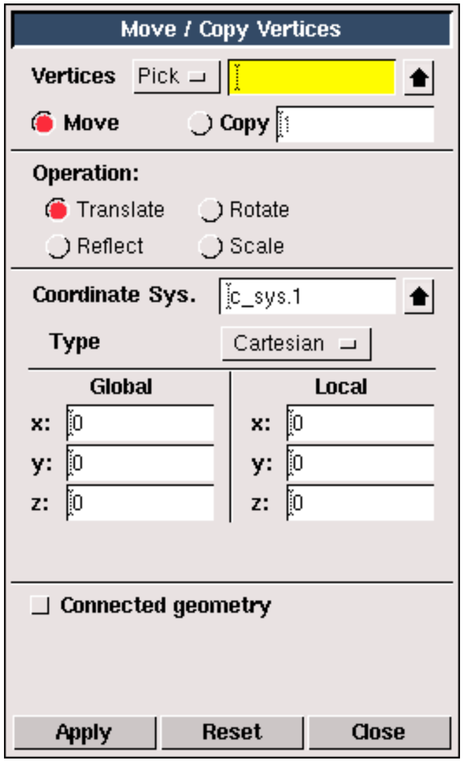


Manipulate Geometry – Scaling

Geometrical scaling of a volume (isotropic)





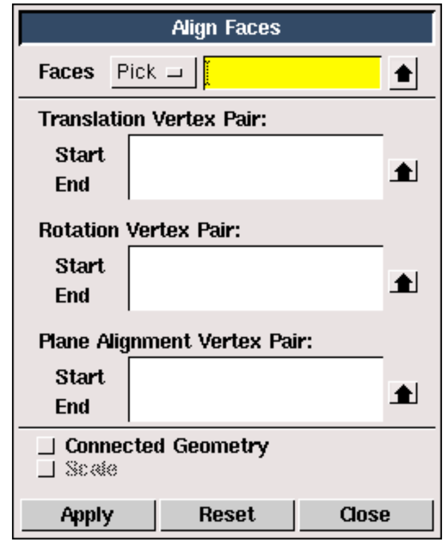


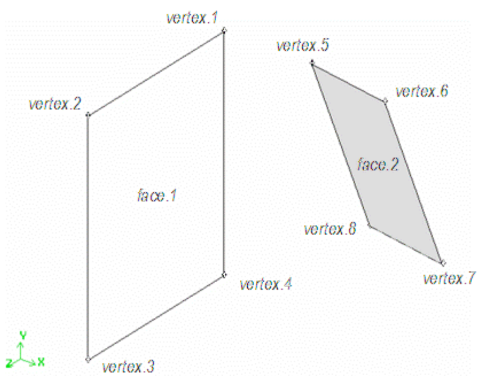
Scaling is based on a Reference Point

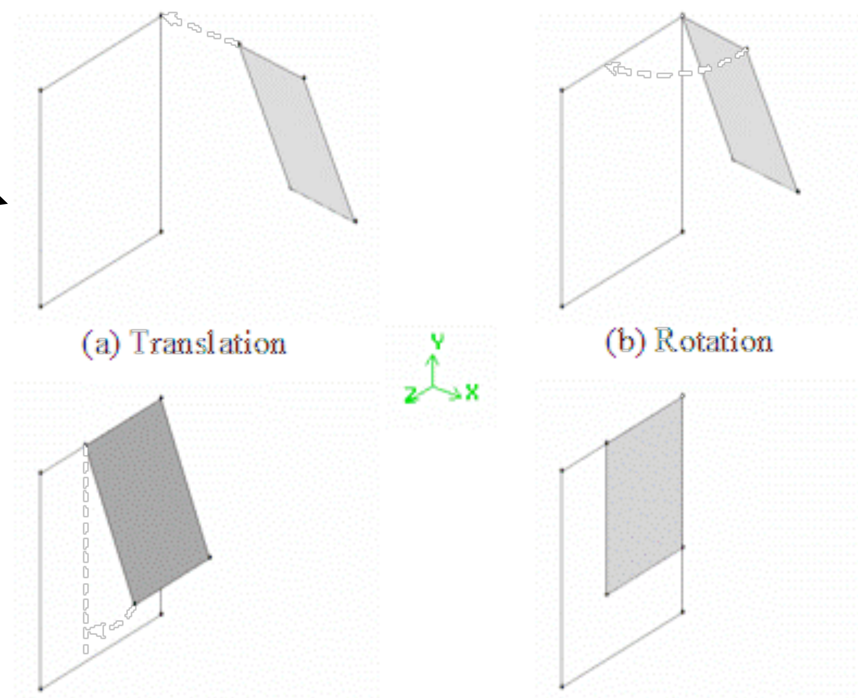
- Default (0,0,0) - origin of the original Cartesian coordinate system
- It is possible to introduce additional coordinate systems

Manipulate Geometry – Align

Modify the geometry of an entity with reference to another one







(a) Translation

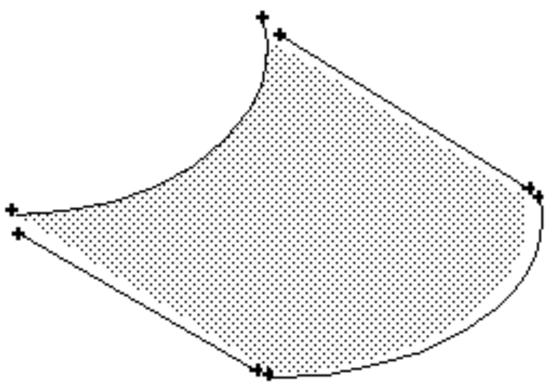
(b) Rotation

(c) Plane alignment

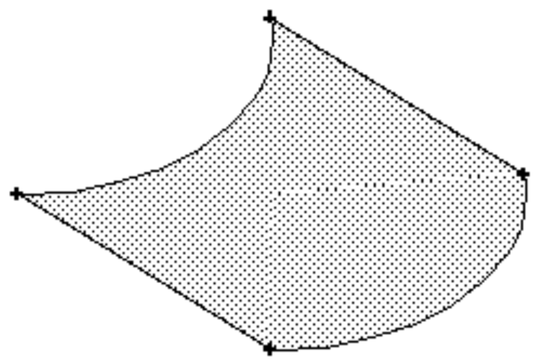
(d) Final configuration

Connect Geometry

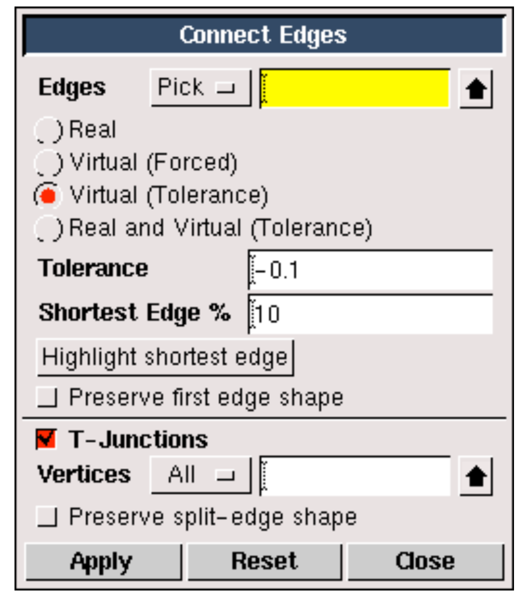
Building upper topological entities from the lower ones requires that they are properly connected



Inconsistent connections



Consistent connections



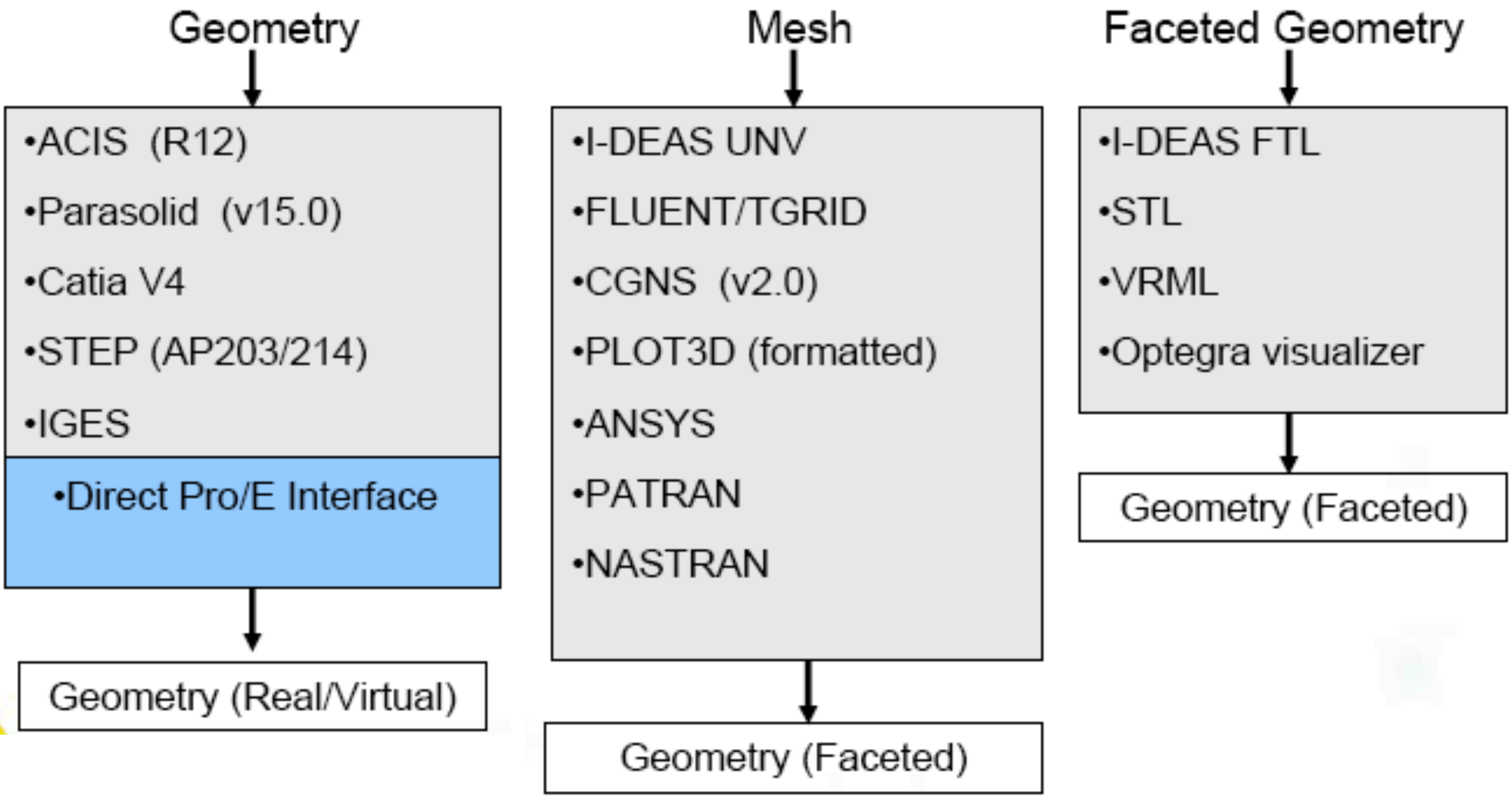
Import Geometries

- Realistic geometries are TOO complicated to be generated from "simple" shapes
- Engineering design is based on CAD systems

<u>Translation between CAD and CFD system is a major bottleneck</u>

- Gambit is based on ACIS geometrical libraries
  - ACIS (Andy, Charles & Ian's System) is the most widely used 3D modeling software technology (http://www.spatial.com)
- It can also import:
  - STEP (STandard for Exchange of Product model data; ISO standard)
  - IGES (Initial Graphics Exchange Specification; ANSI standard)
  - STL (STereo Lithography; Rapid Prototyping Standard)
  - ….

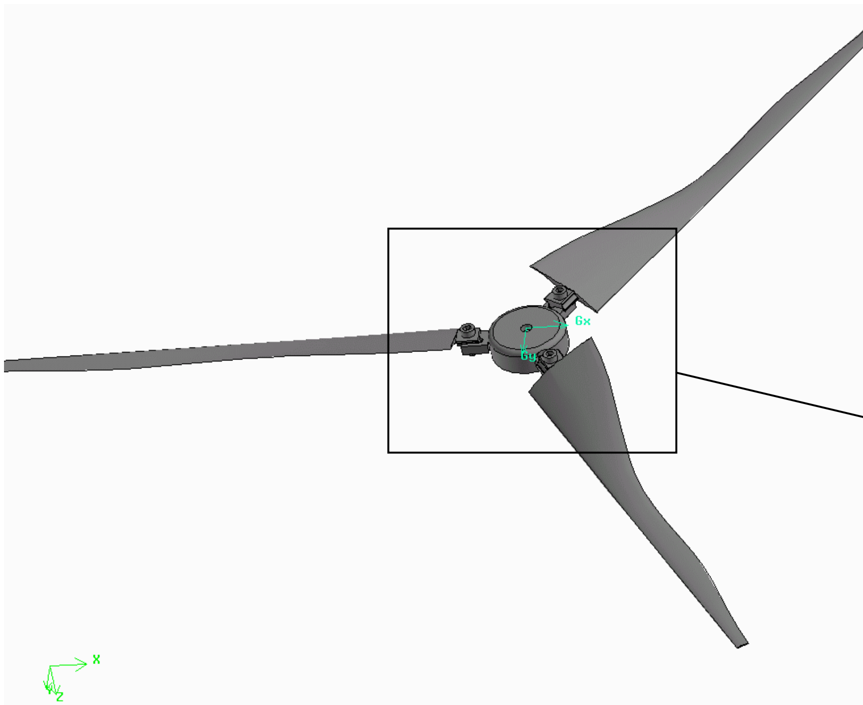
Import Geometries

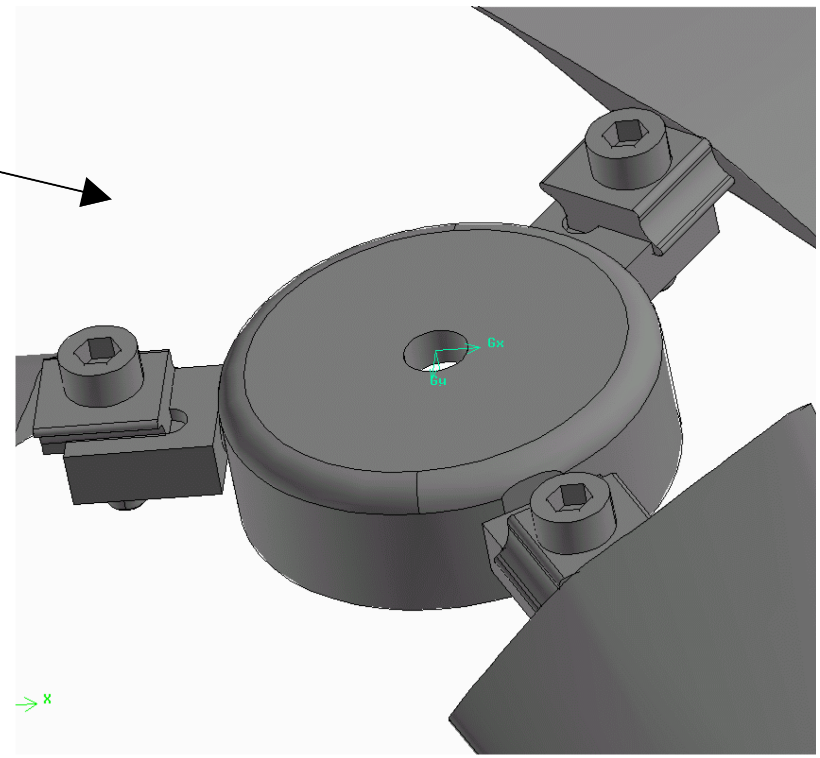


Clean-Up a CAD Model

- Eliminate components not exposed to the flow
- Eliminate duplicated entities
- Eliminate small details
- Water-proofing the surfaces
- Rebuild geometrical connectivity between parts

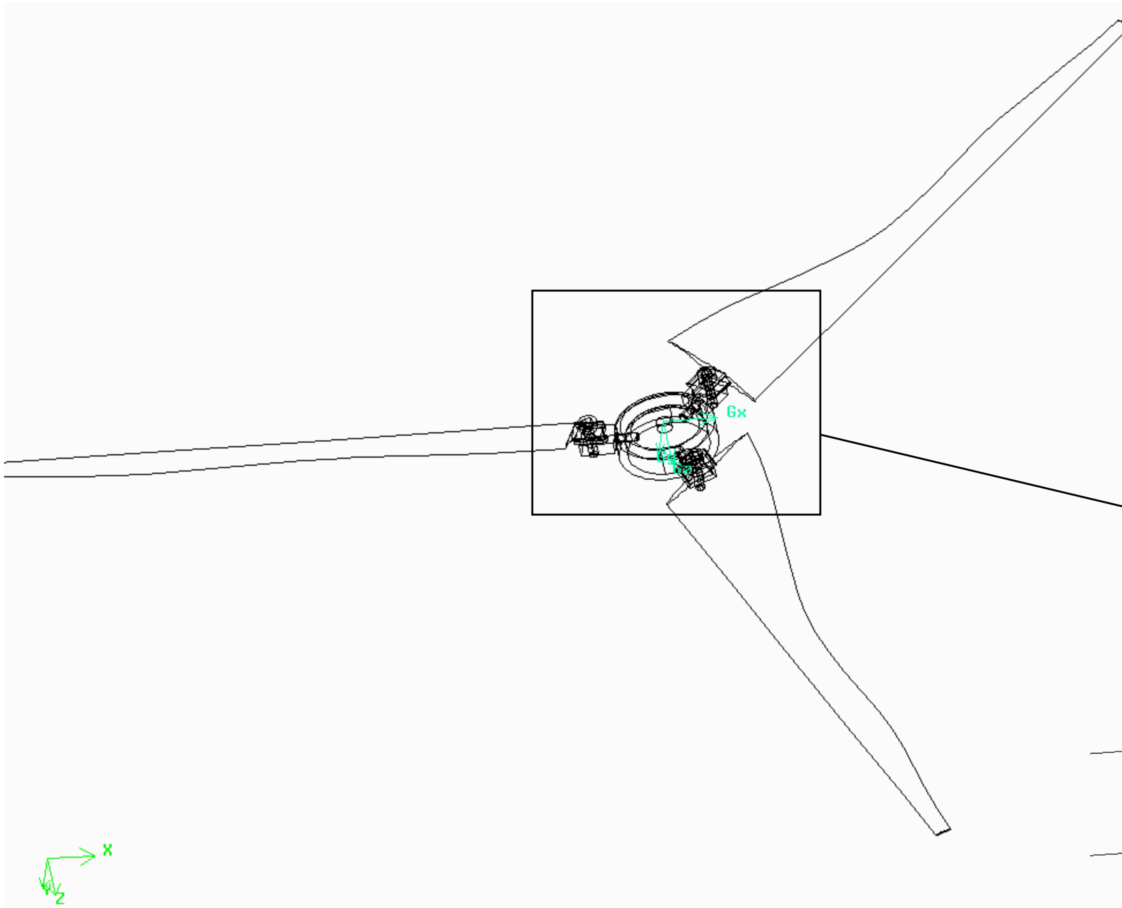
Example: Helicopter Rotor

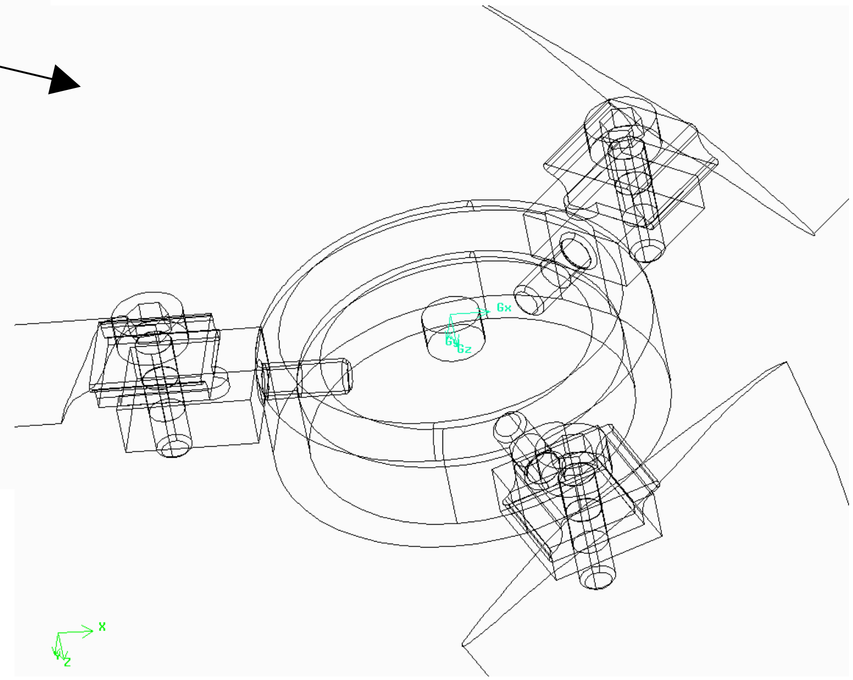




The rotor-shaft connection
is VERY complicated

Geometrical entities





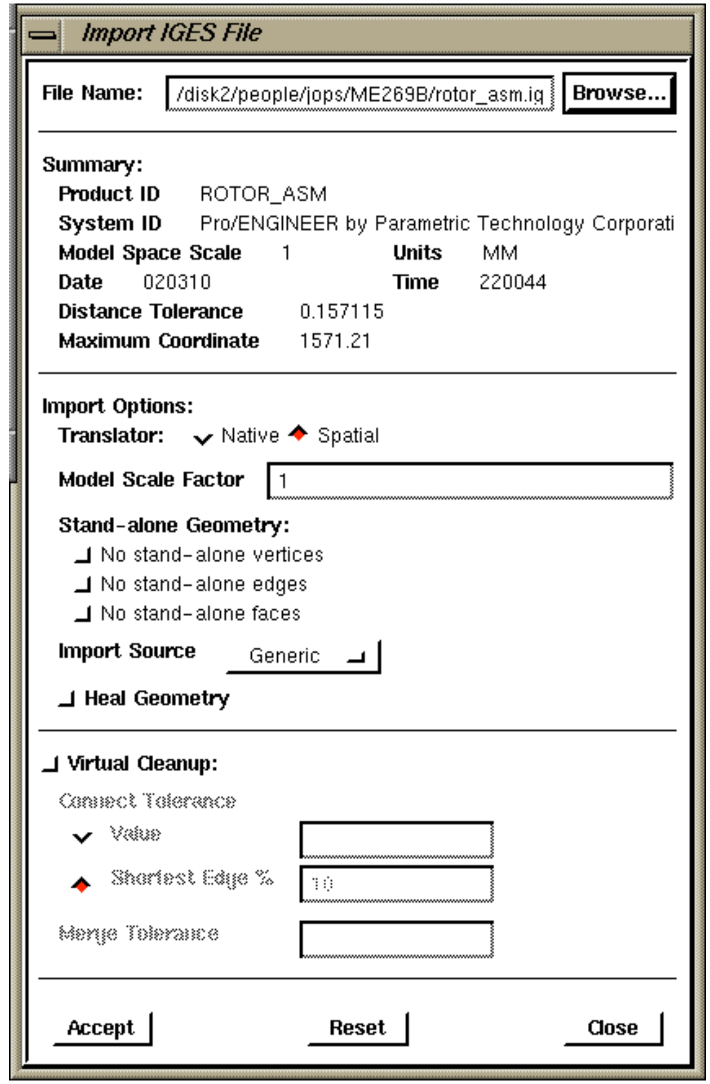
The geometry consists of 10 components
3 blades, 1 shaft support, 6 connectors

Example: Import IGES Model

IGES export is available from every CAD system

IGES models are a collection of "untrimmed" edges and faces

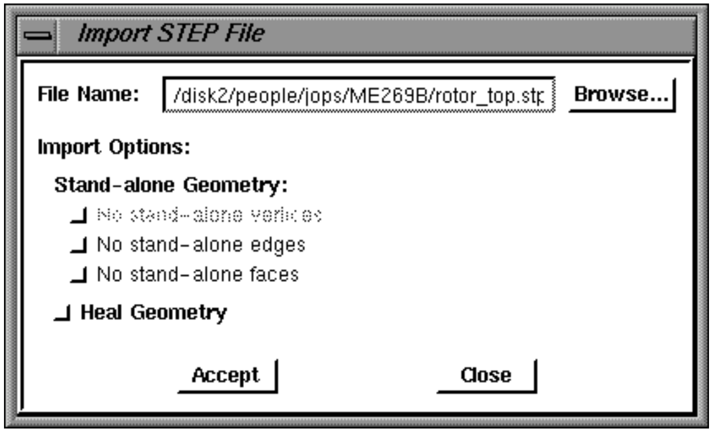
Imported geometry consists of:
0 volumes
~250 faces
~1100 edges
~1000 vertices



Example: Import STEP Model

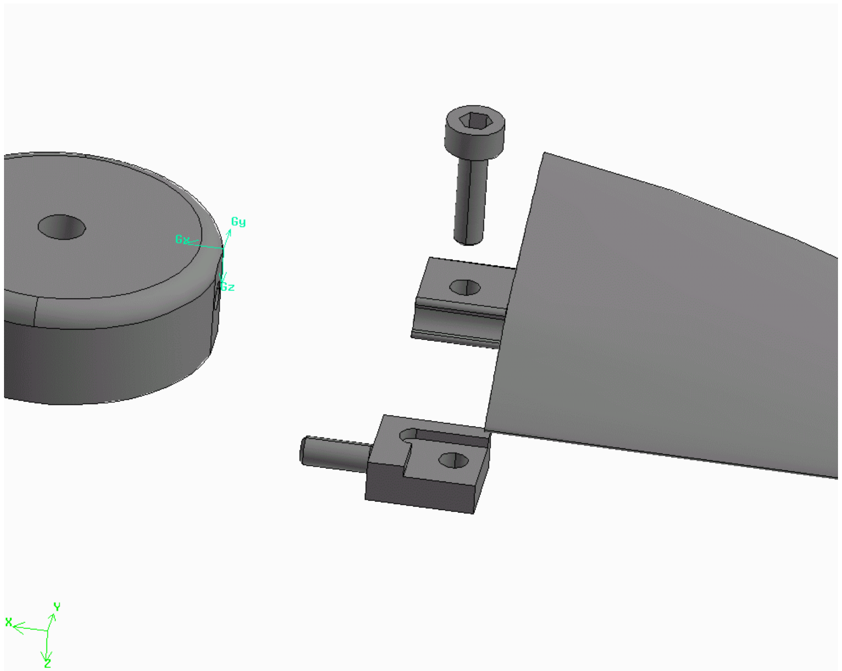
STEP export is available from many CAD system

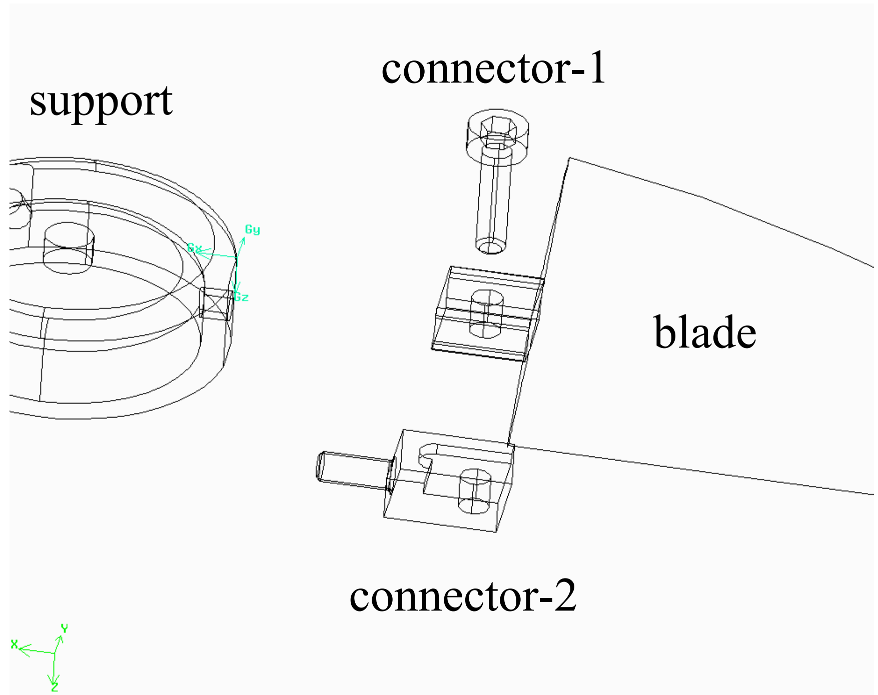
STEP models are a collection of parts or components



Imported geometry consists of:
10 volumes
~190 faces
~450 edges
~300 vertices

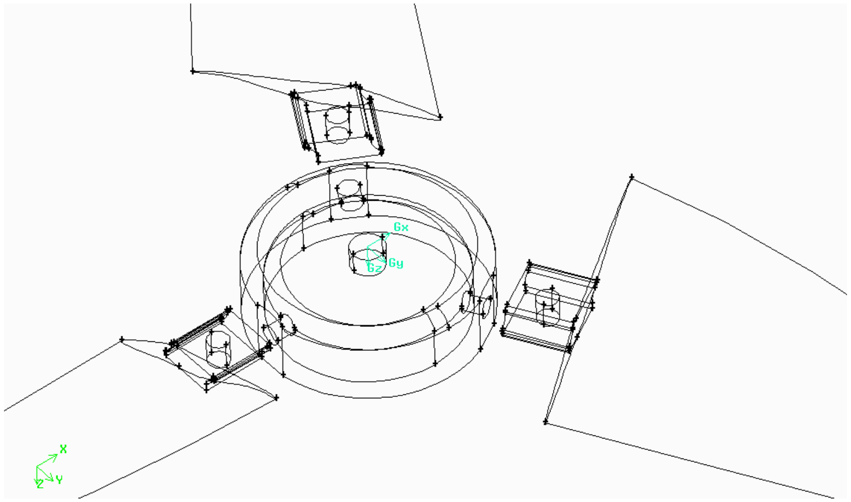
Components "exploded"



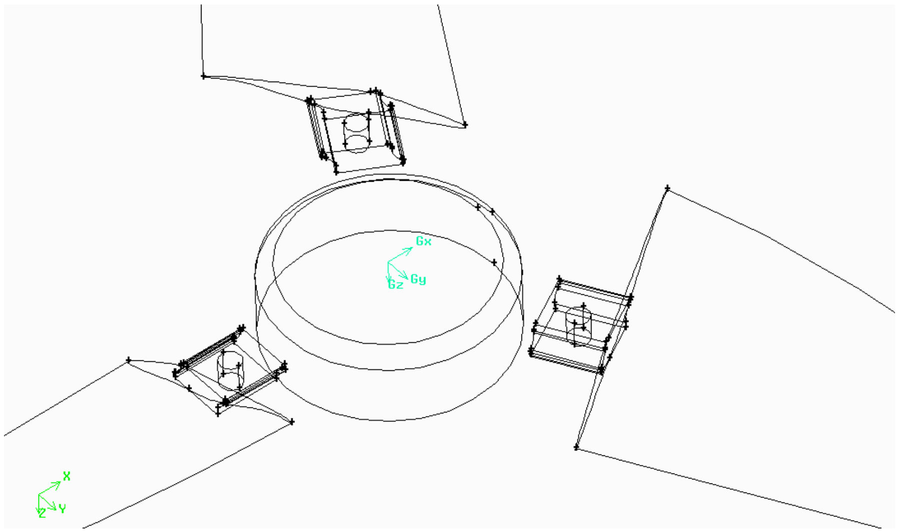


For aerodynamic analysis the details of the rotor-shaft connection are not important. The geometry of the blades MUST be preserved

Geometry simplification

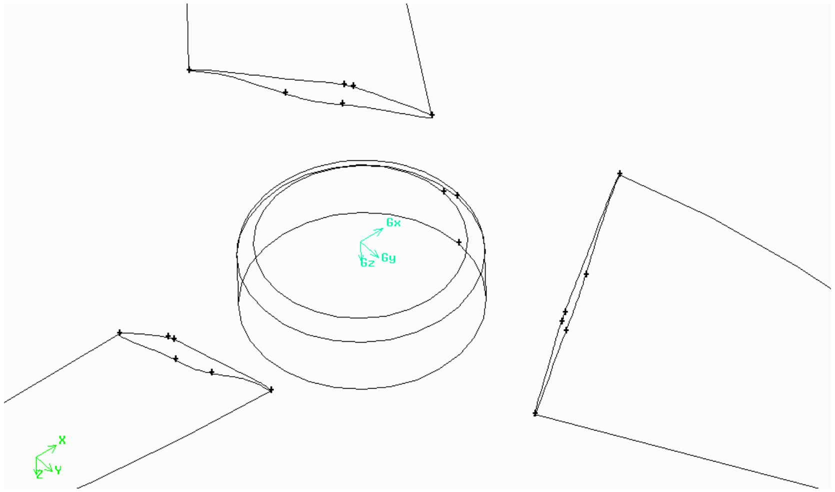


Connectors eliminated

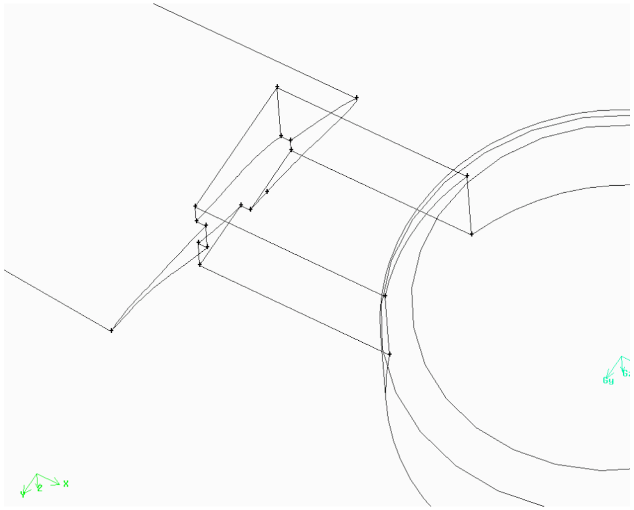


Support generated as a "simple" cylinder with blended side

Geometry simplification

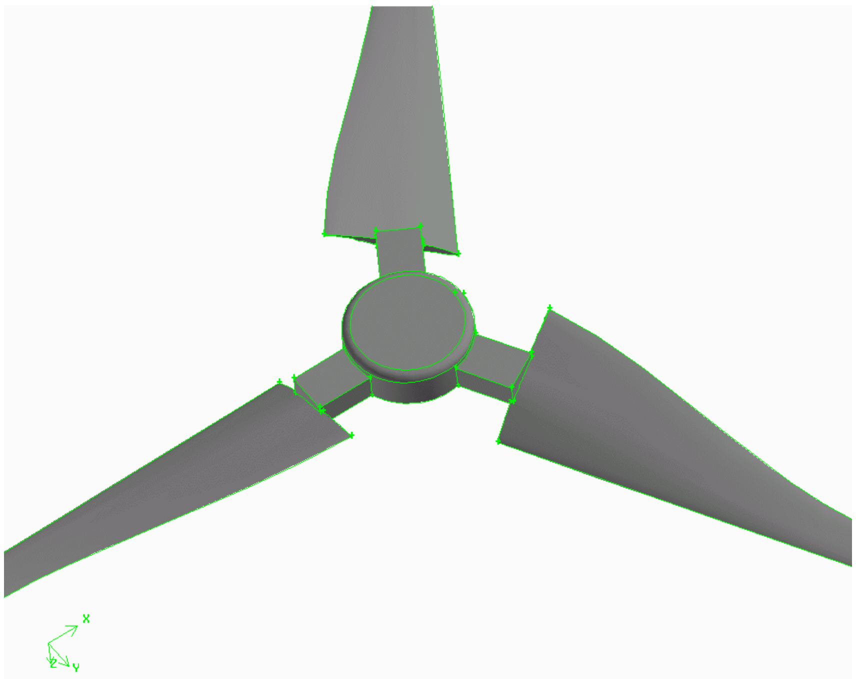


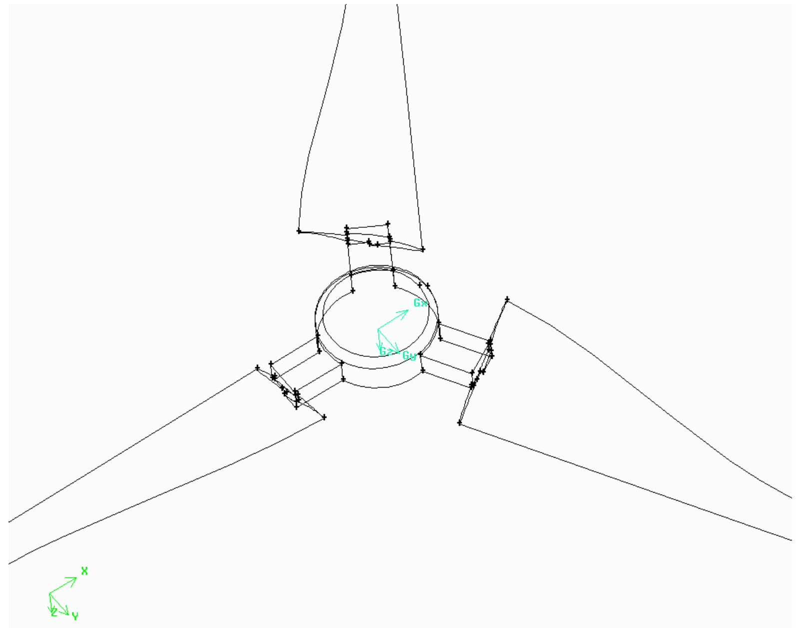
Blade edge cleaned and *sealed*



Blade-support connector is a “simple” cuboid

Clean Geometry





This example is available on the class Web site

Virtual Geometry

GAMBIT operates on two different type of entities

REAL: with corresponding geometrical and topological characteristics
VIRTUAL: defined *only* with reference to REAL or other VIRTUAL entities

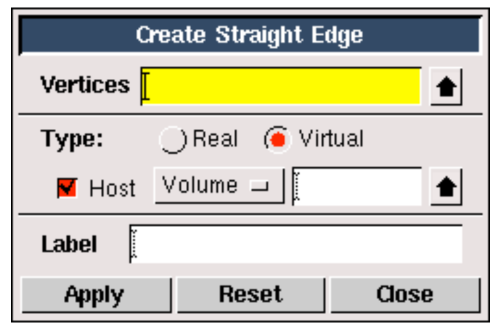
REAL entities are what we used and described so far;
VIRTUAL are used to SIMPLIFY, CLEAN UP, DECOMPOSE real entities

Note that some geometry tools cannot be applied to virtual entities (boolean operation, volume blending, creation of volumes by sweeping faces, etc.)

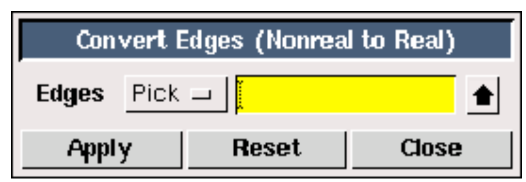
Virtual Geometry

Superset: Entity that references two or more real entities
Interpolant: Entity represents an average/interpolant ot various real entities
Parasite: Entity defined completely from a real entity

The virtual geometry is typically constructed using a host entity

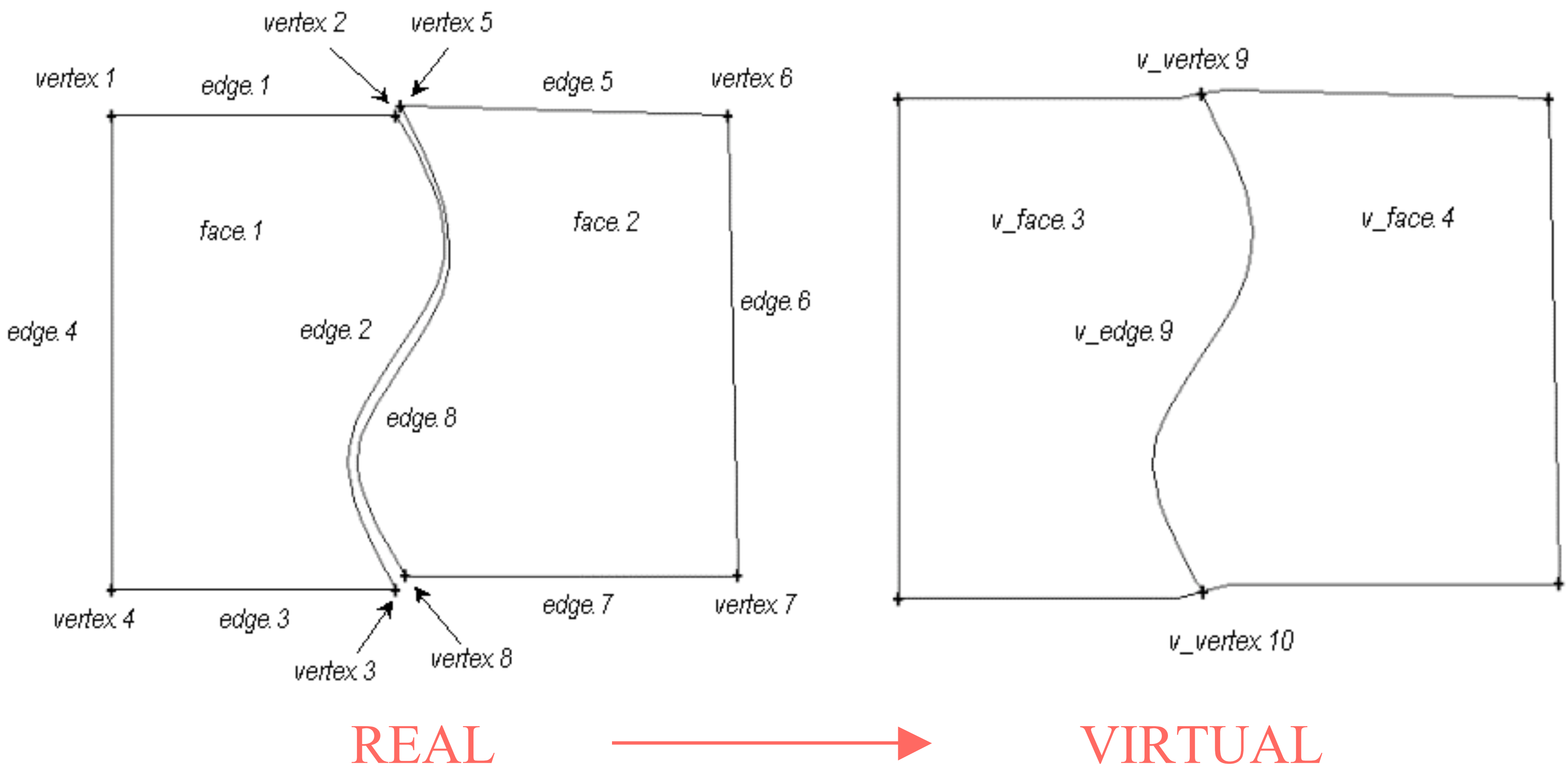


Real entities can be transformed in virtual but NOT ALWAYS viceversa



Virtual Geometry Clean-Up

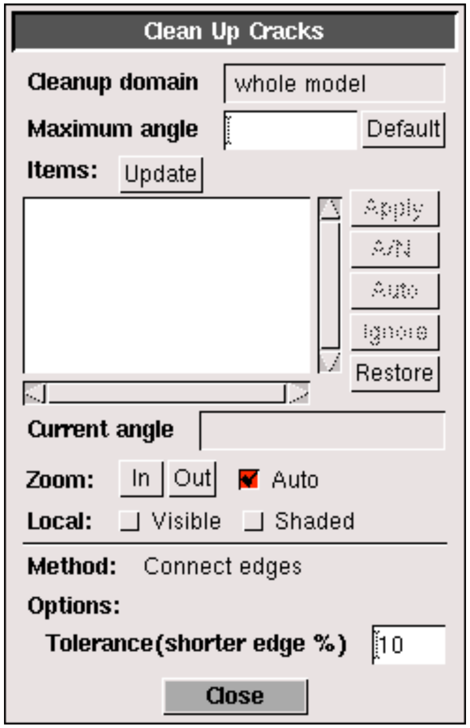
Example of edge connecting operation: virtual interpolant

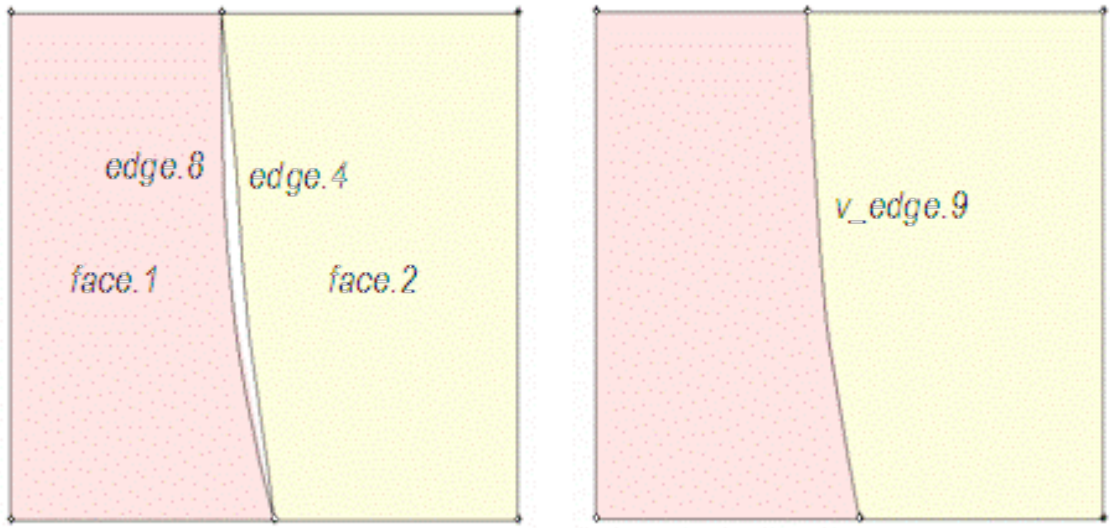


Clean-Up Cracks

A "crack" is defined as a geometry consisting of an edge pair that meets the following criteria.

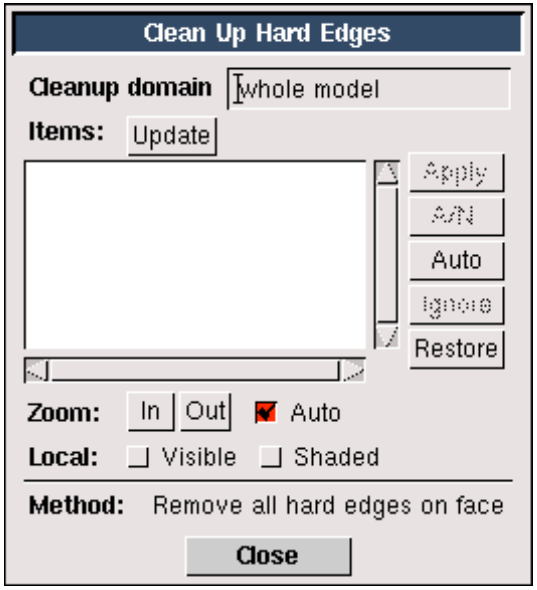
- Each edge in the pair serves as a boundary edge for a separate face.
- The edges share common endpoint vertices at one or both ends.
- The edges are separated along their lengths by a small gap.





(a) Before cleanup (b) After cleanup

Clean-Up Hard Edges



Hard edges (dangling edges) are those that are included in the list of edges that define a face but which do not constitute necessary parts of the closed edge loop that circumscribes the face.

Such edges often result from face-split operations in which the split-tool face only partially intersects the target face.



(a) Before cleanup (b) After cleanup

Grid Generation

- Geometry definition (simple shapes, CAD import)
- Grid generation algorithms
- GAMBIT
- Grid quality and improvement
- Automation

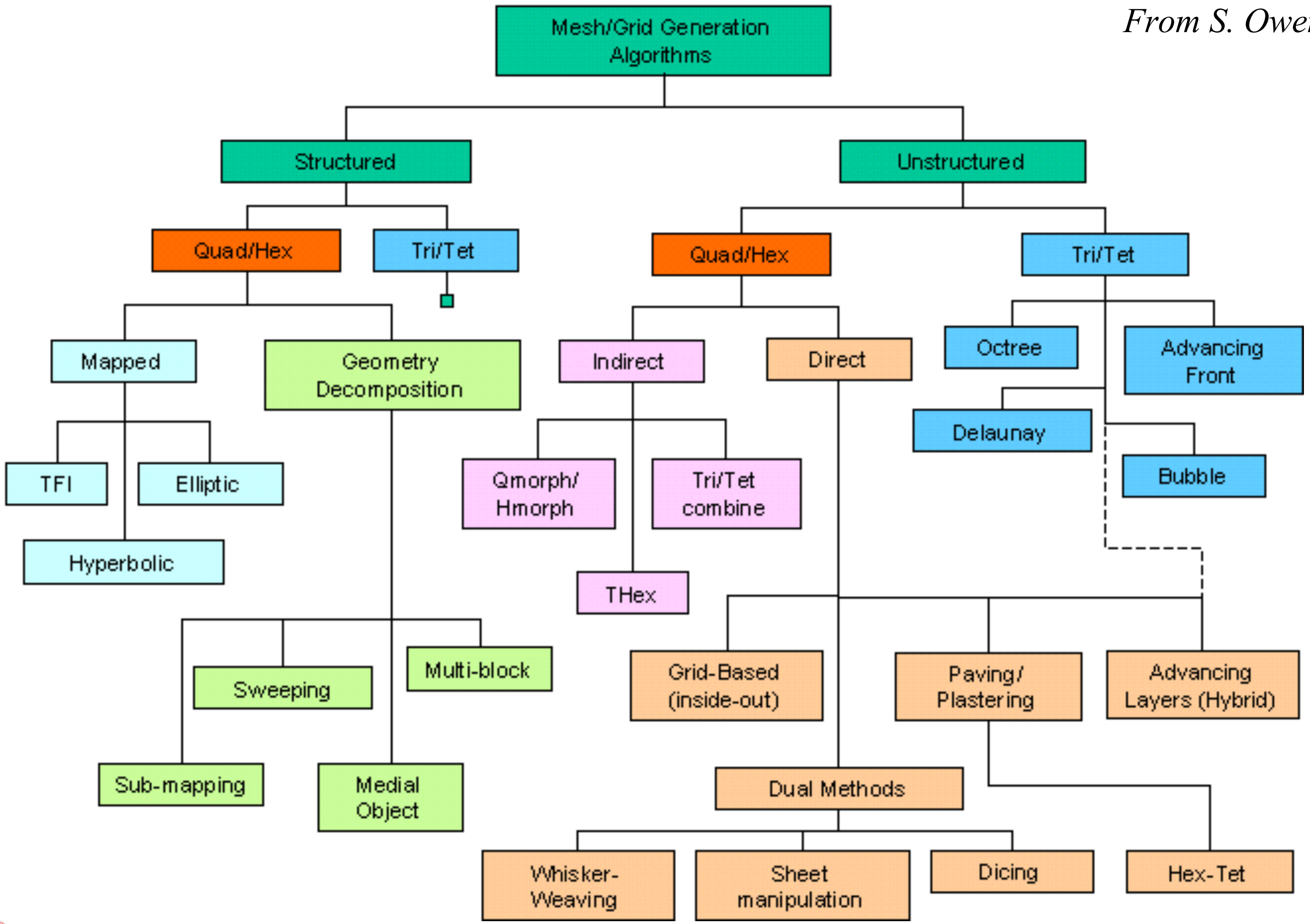
Grid generation techniques

- Structured grids
  - Ordered set of (locally orthogonal) lines
  - Several Techniques can be used to Map a computational domain into a physical domain: Transfinite Interpolation, Morphing, PDE Based, etc.
  - The grid lines are curved to fit the shape of the boundaries

- Unstructured grids
  - Unorganized collection of polygons (polyhedron)
  - Three main techniques are available to generate automatically triangles (tetrahedra): Delaunay triangulation, Advancing front, OCTREE
  - Paving for automatic generation of quads in 2D

Grid generation techniques

From S. Owen, 2005



Grid generation techniques

Gambit is a "commercial" grid generator and includes only few (relatively standard) algorithms. New methods are slow to gain robustness and generality and therefore are not directly available

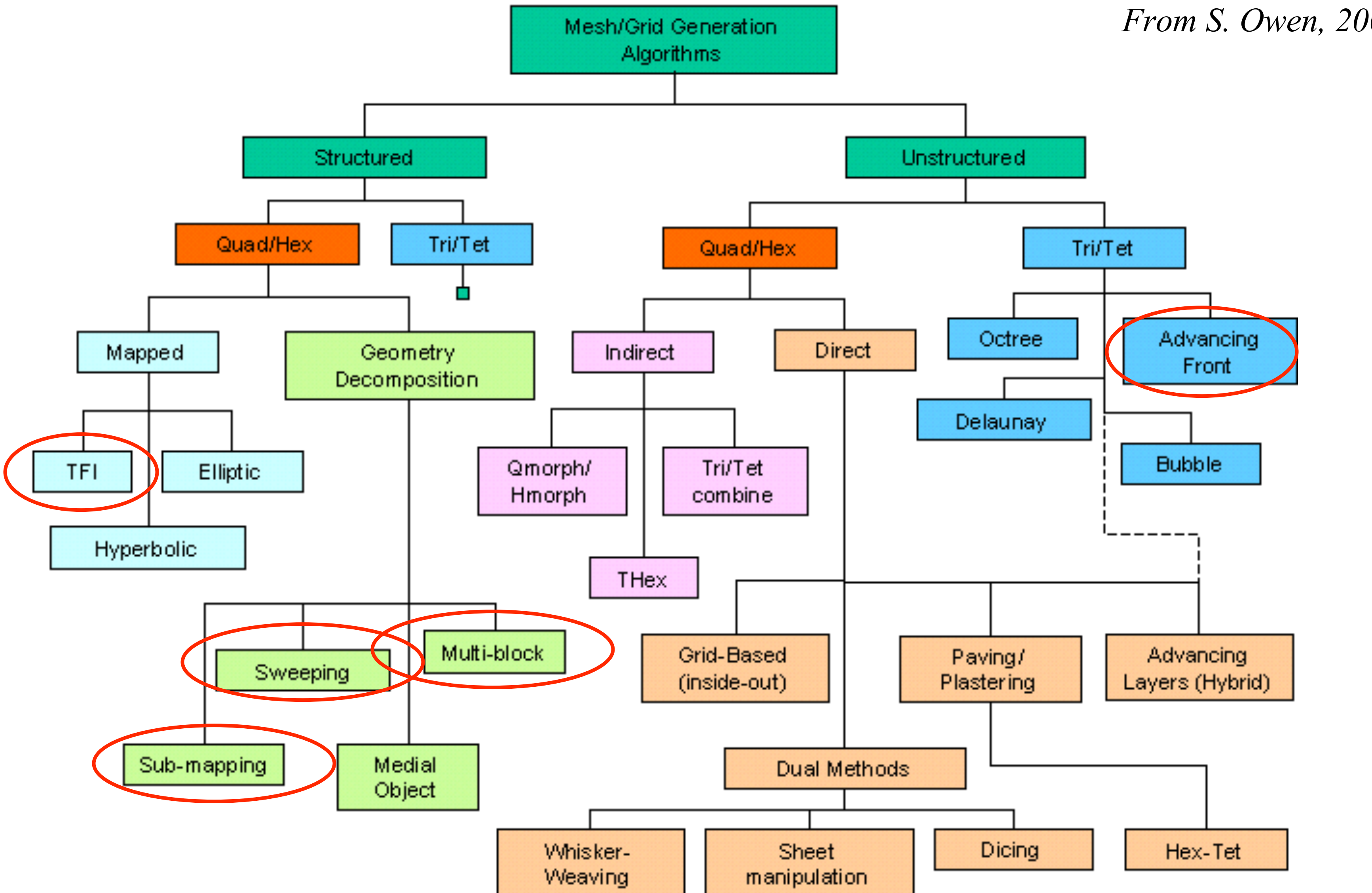
Cubit is a "research" grid generator and the latest approaches are typically included (several of them have been actually invented by the Cubit team)



http://cubit.sandia.gov/

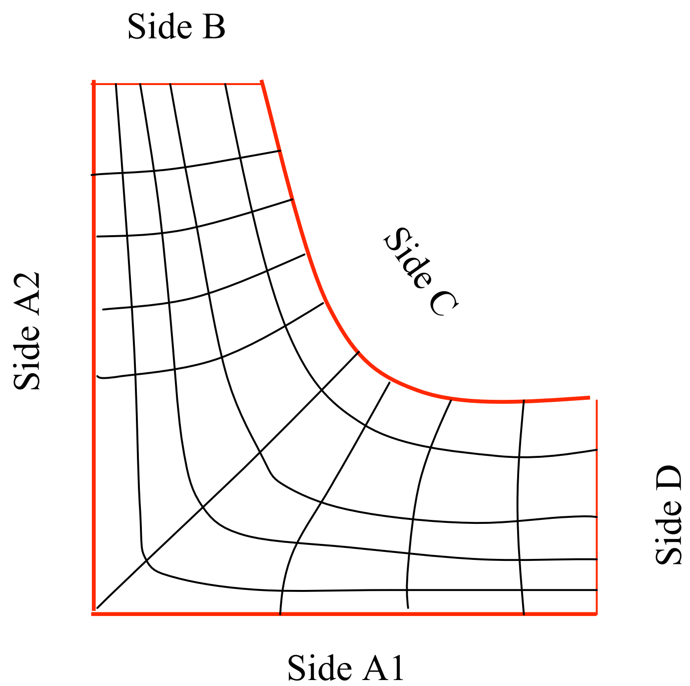
Grid generation techniques

From S. Owen, 2005

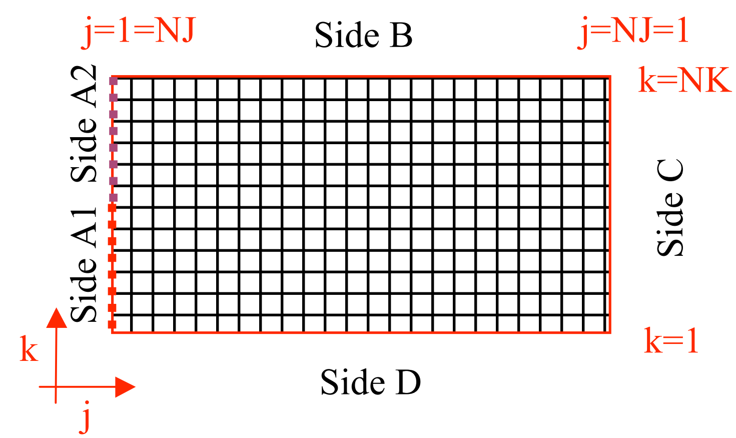


Structured Grids: Mapping

Transfinite Interpolation



Physical Domain



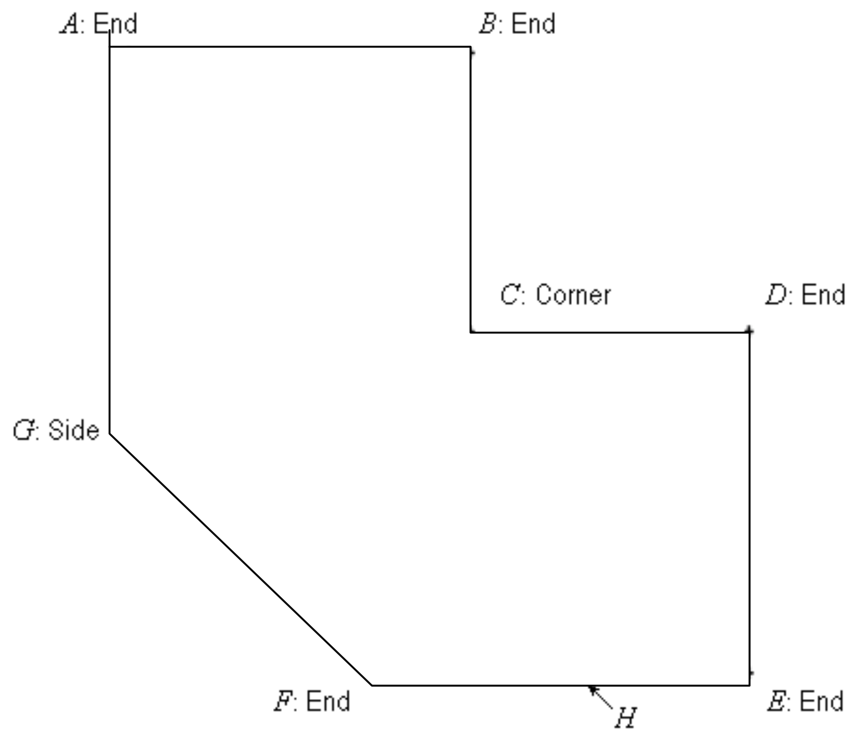
Computational Domain

Structured Grids: Sub-mapping

Regions are automatically subdivided in “mappable areas”

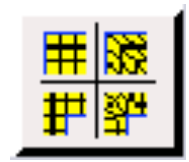
Number of grid elements
have to be chosen consistently

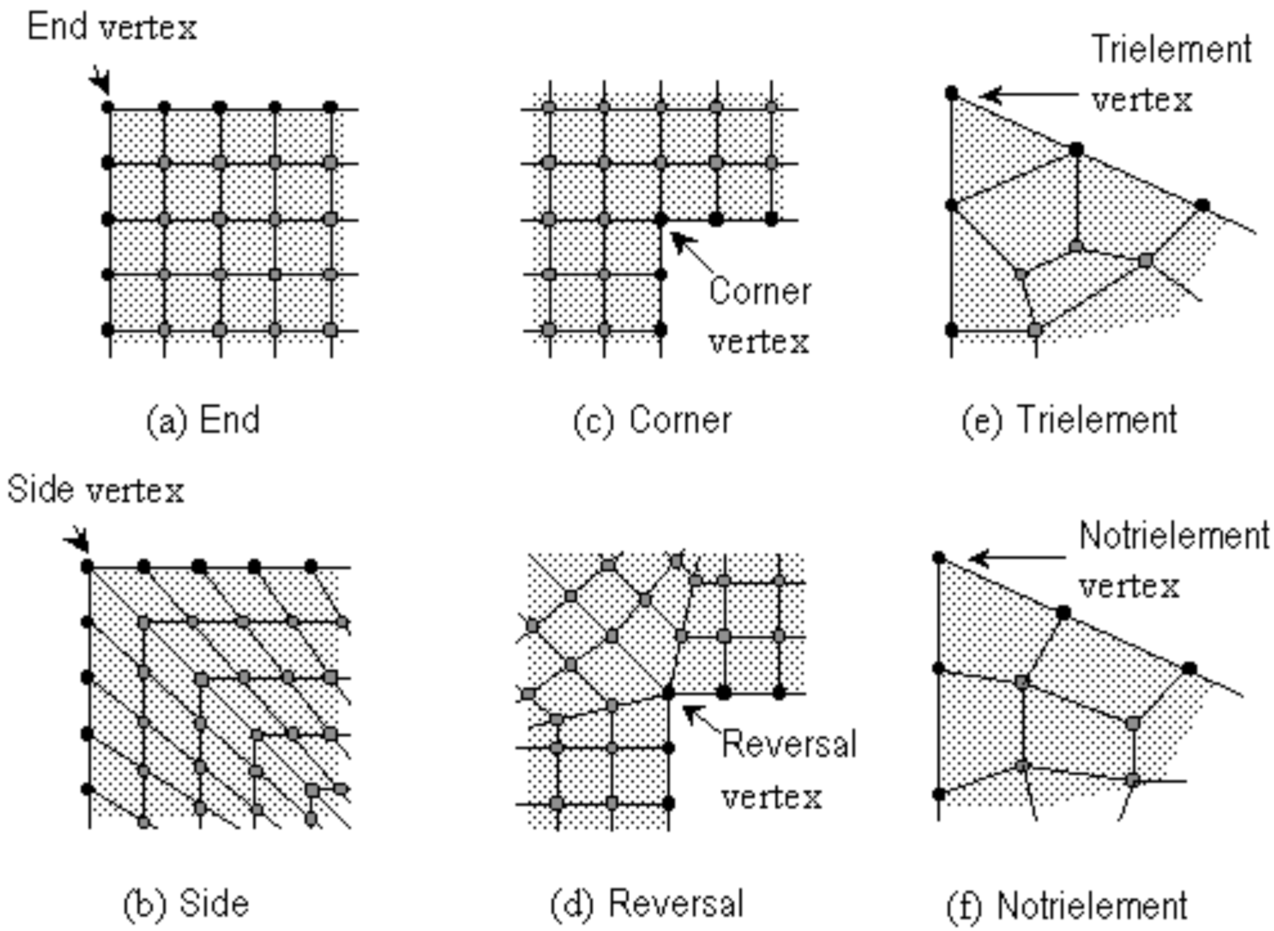
The grid type is controlled
by a vertex attribute



Vertex-face type

User can specify the behavior of the grid at a certain node

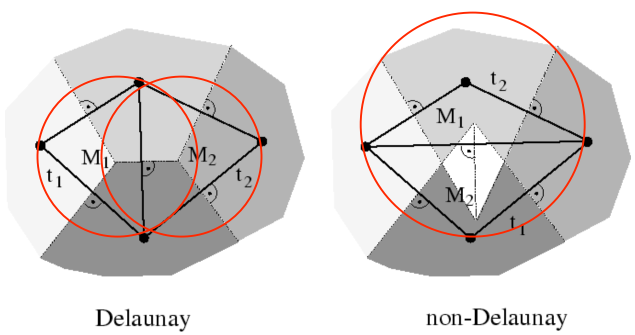


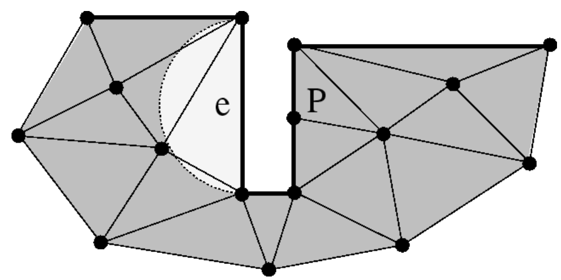


Unstructured Grids: Triangulations

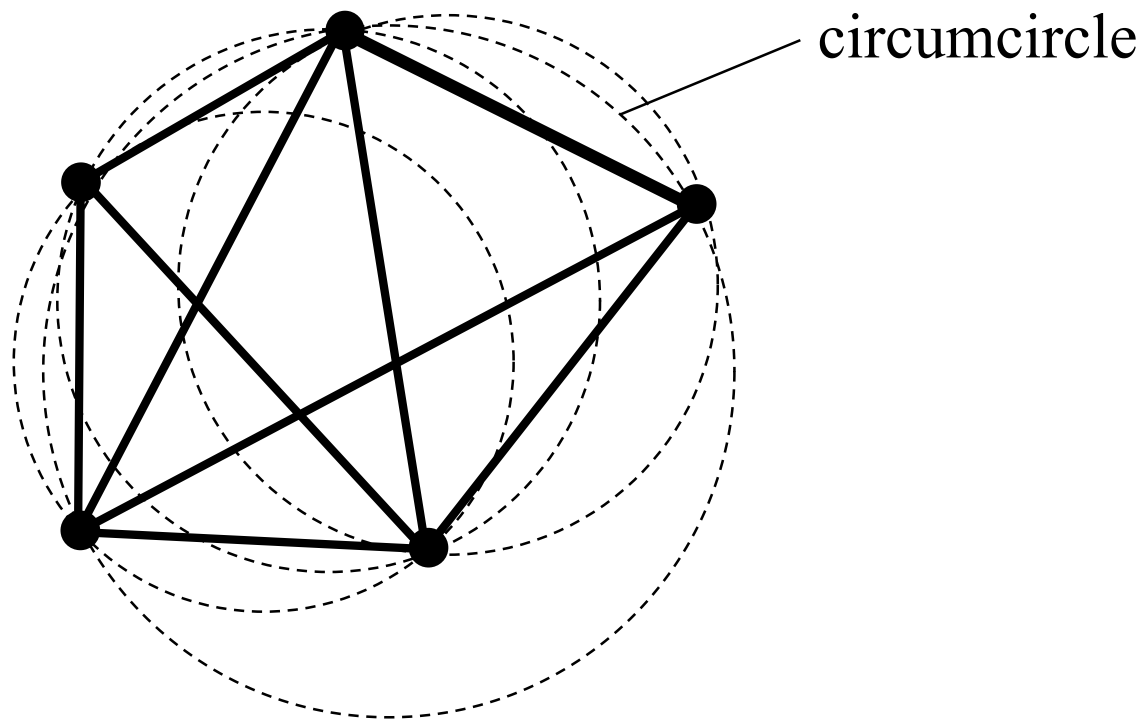
- Delaunay
  - Empty circle principle: *any node must not be contained within the circumcircle (circle passing through the vertices of a triangle) on any triangle within the mesh*
  - Automatic triangulation of random set of nodes
  - Nodes are inserted locally in a triangulation and triangles are redefined locally to satisfy the Delaunay criterion (available mathematical tools)

\+ Inherent grid quality
\+ Elegant mathematical basis
\- Boundary integrity



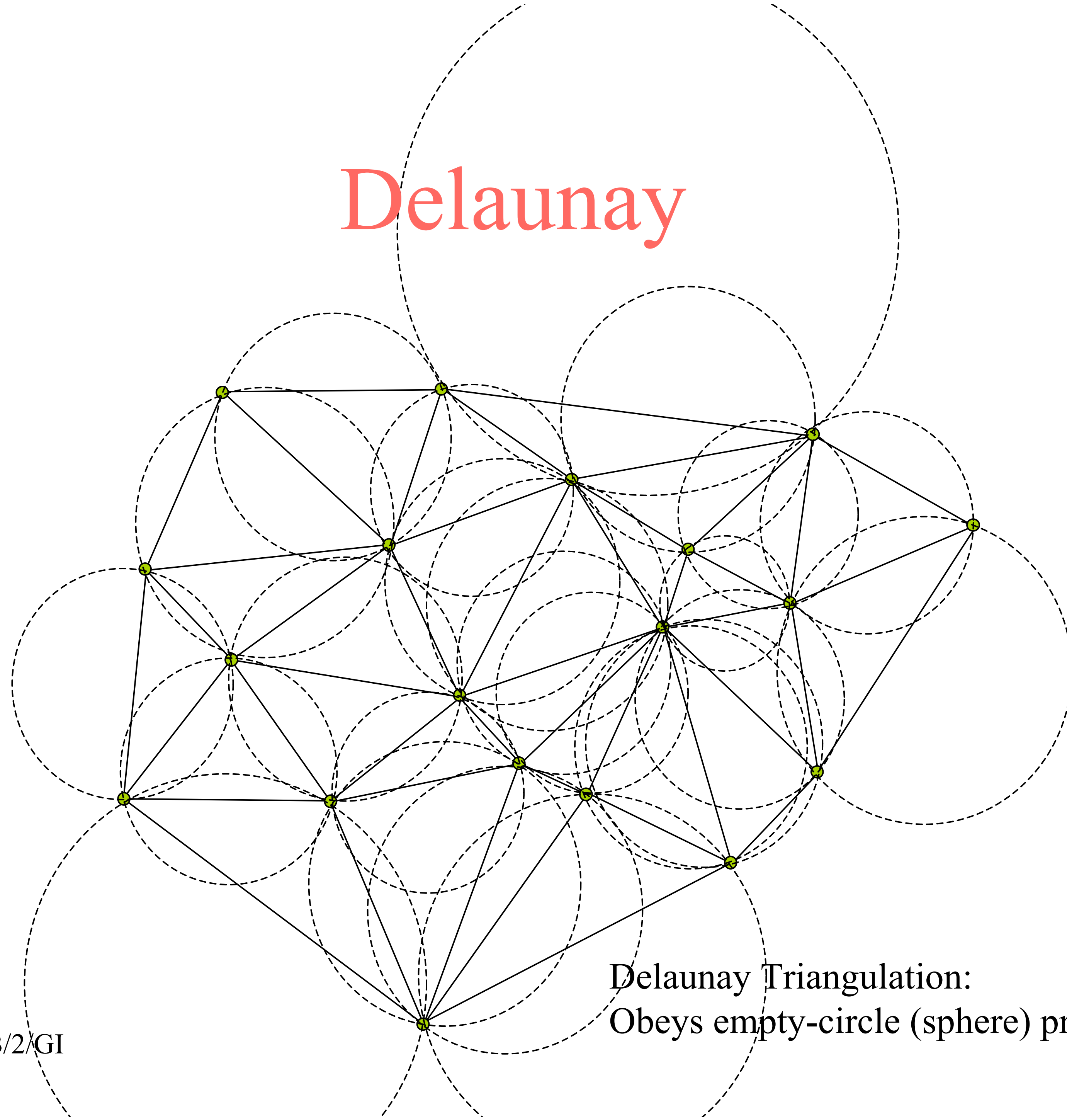


Delaunay



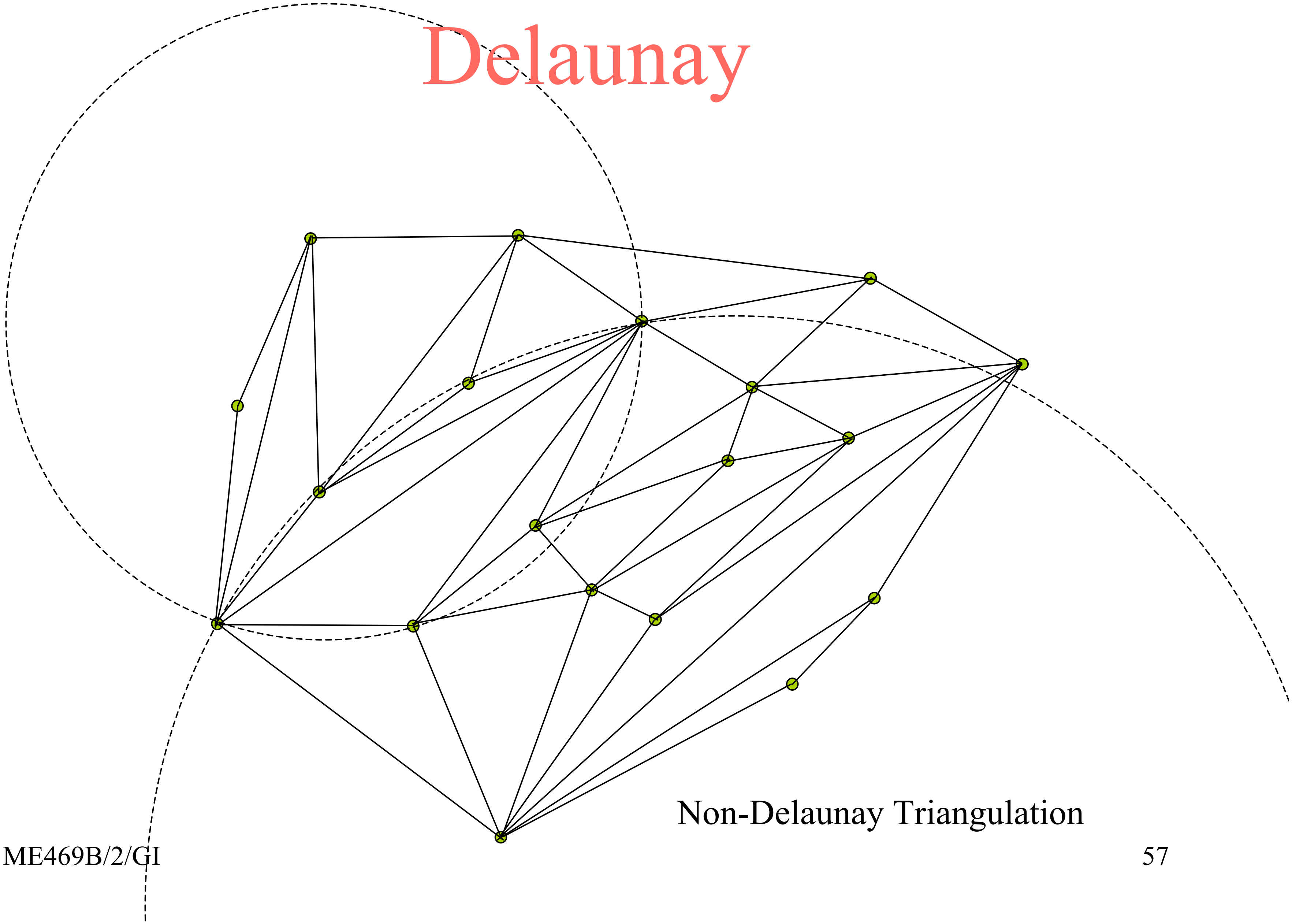
Empty Circle (Sphere) Property:
No other vertex is contained within the circumcircle (circumsphere) of any triangle (tetrahedron)

Delaunay



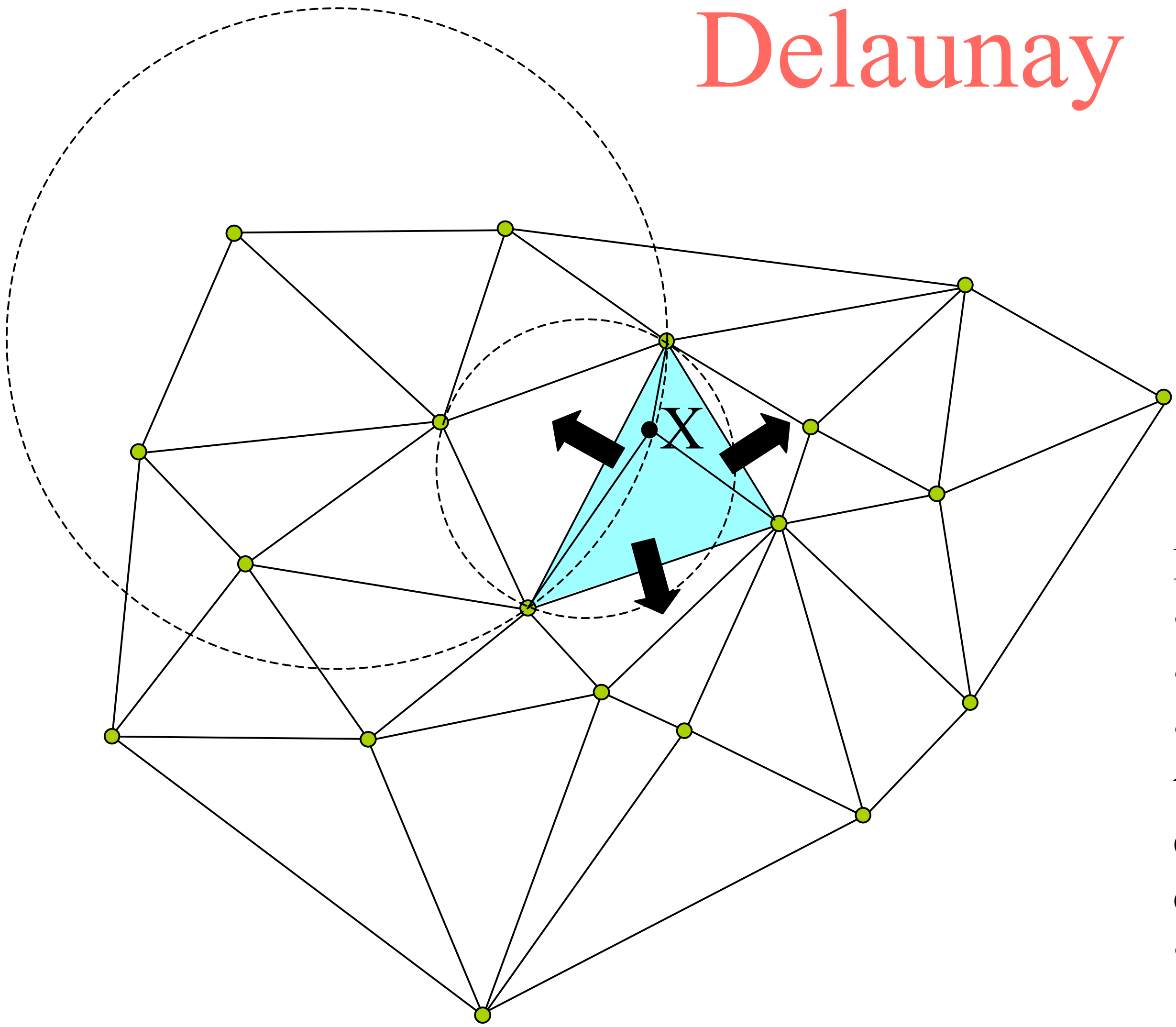
Delaunay Triangulation:
Obeys empty-circle (sphere) property

Delaunay



Non-Delaunay Triangulation

Delaunay

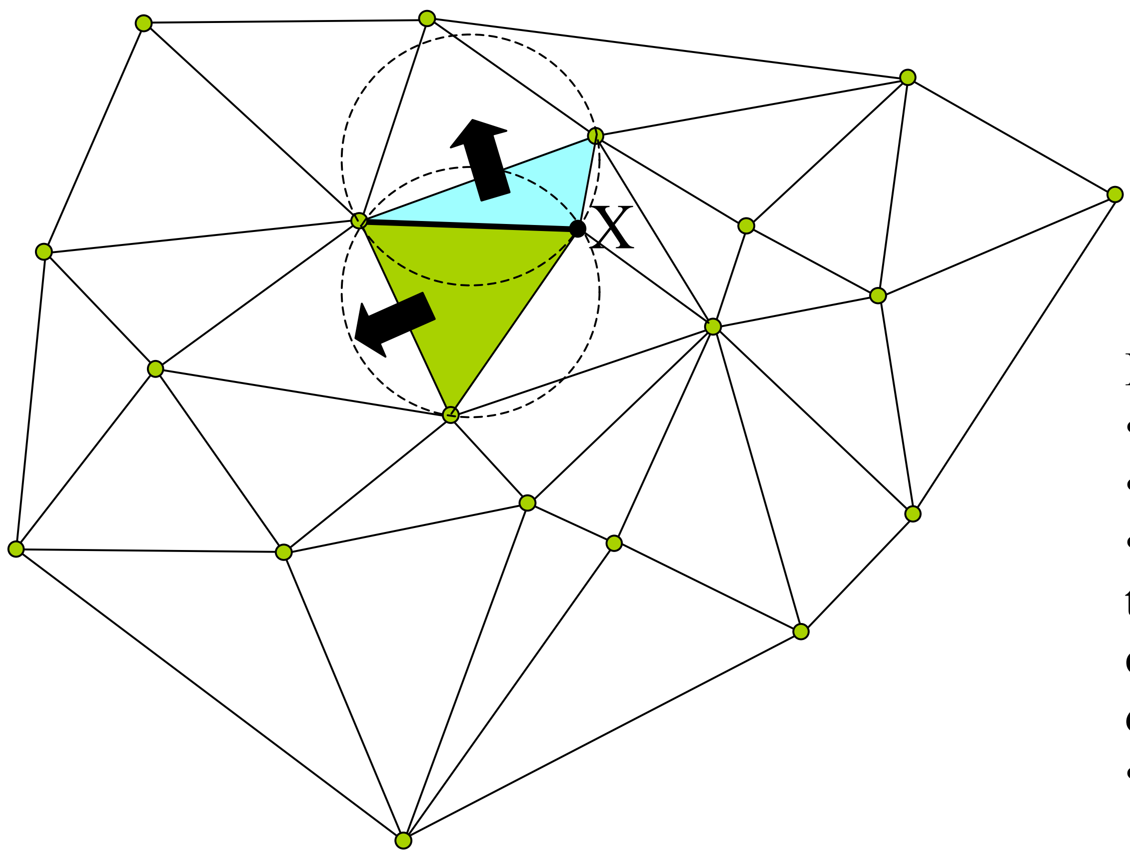


Given a Delaunay Triangulation of n nodes, How do I insert node n+1 ?

Lawson Algorithm

- Locate triangle containing X
- Subdivide triangle
- Recursively check adjoining triangles to ensure empty-circle property. Swap diagonal if needed
- (Lawson,77)

Delaunay



Lawson Algorithm

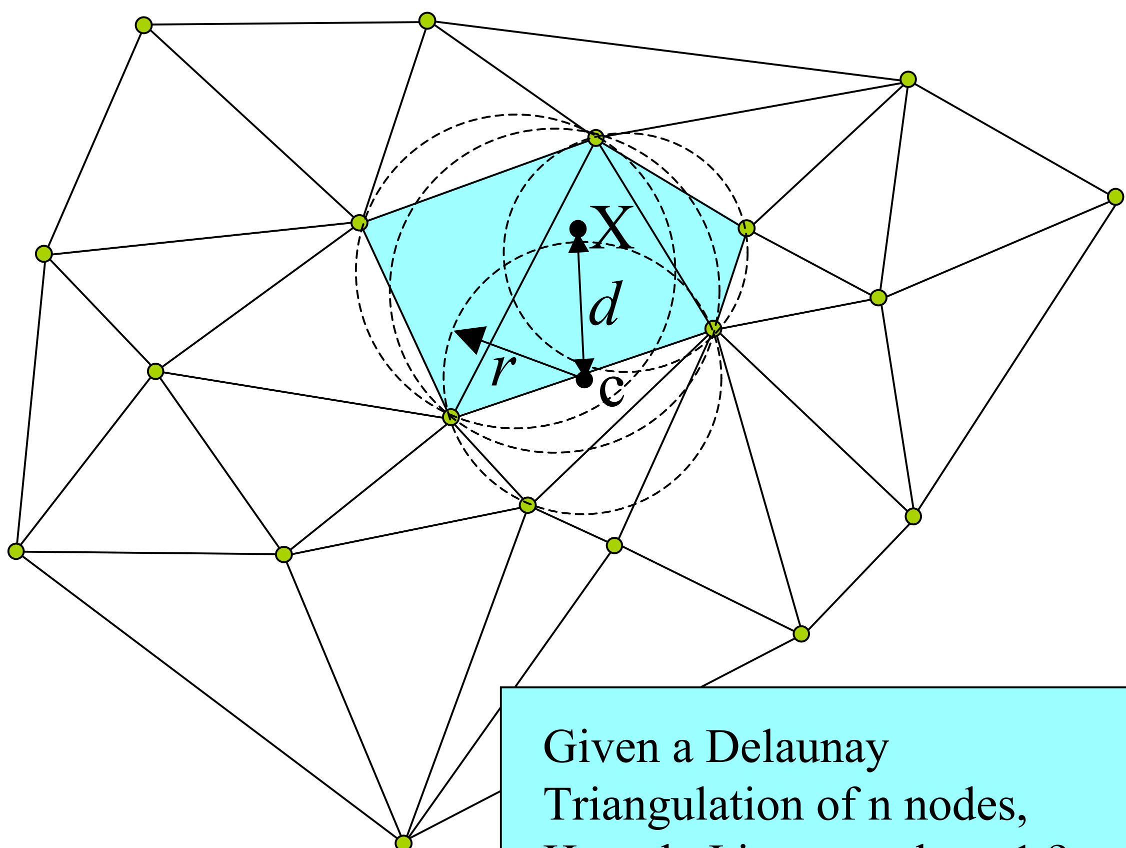
•Locate triangle containing X

•Subdivide triangle

•Recursively check adjoining triangles to ensure empty-circle property. Swap diagonal if needed

•(Lawson,77)

Delaunay

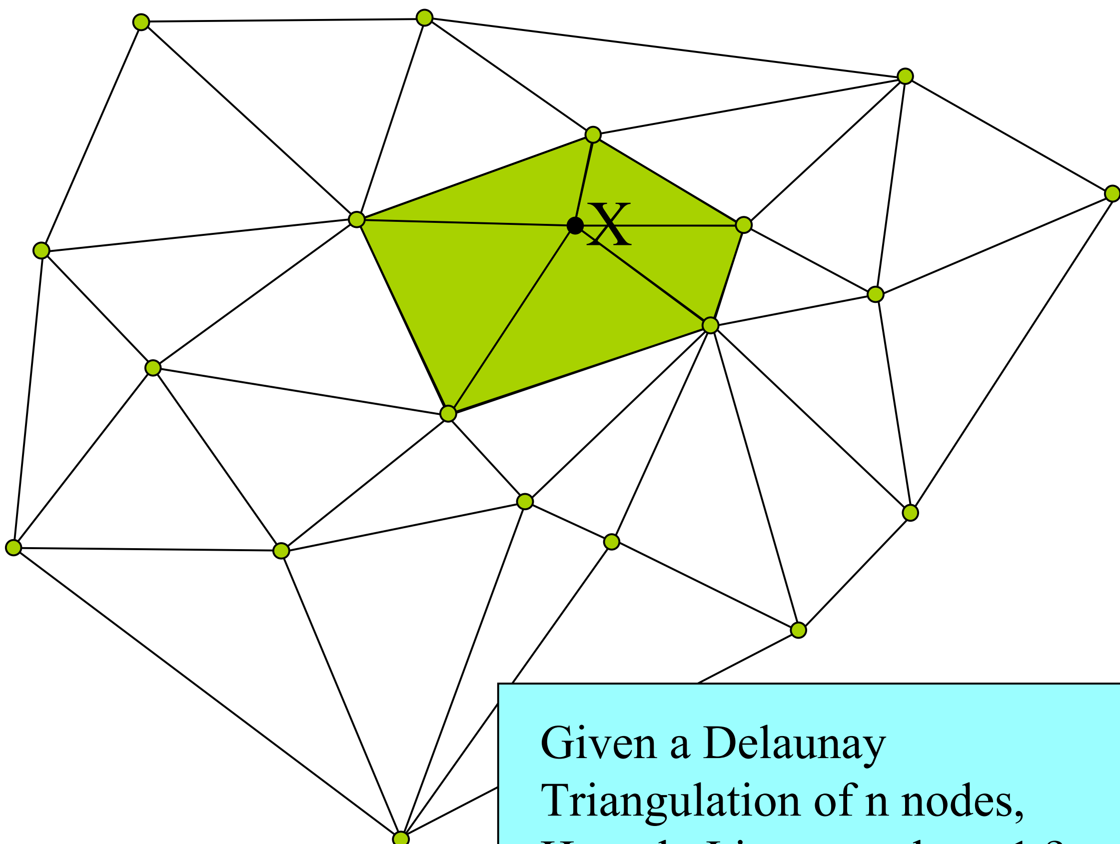


Bowyer-Watson Algorithm

- Locate triangle that contains the point
- Search for all triangles whose circumcircle contain the point ($d<r$)
- Delete the triangles (creating a void in the mesh)
- Form new triangles from the new point and the void boundary
- (Watson,81)

Given a Delaunay Triangulation of n nodes, How do I insert node n+1 ?

Delaunay



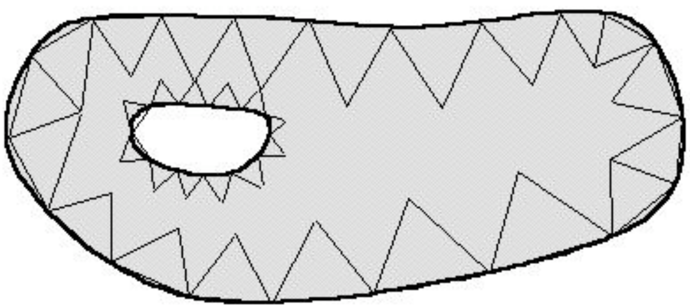
Bowyer-Watson Algorithm
- •Locate triangle that contains the point
- •Search for all triangles whose circumcircle contain the point ($d<r$)
- •Delete the triangles (creating a void in the mesh)
- •Form new triangles from the new point and the void boundary
- •(Watson,81)

Given a Delaunay Triangulation of n nodes, How do I insert node n+1 ?

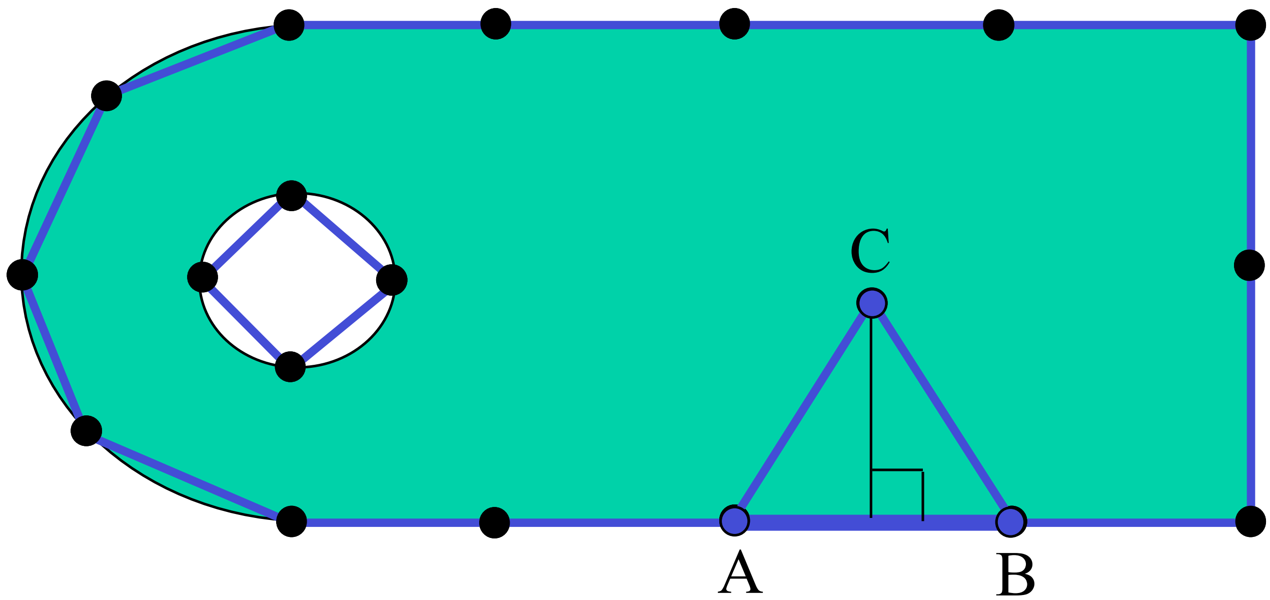
Unstructured Grids: Triangulations

- Advancing front
  - Triangles are built inward from the boundary surfaces
  - The last layer of elements constitutes the active front
  - An optimal location for a new nodes is generated for each segment on the front; the new node is generated by checking all existing nodes and this new optimal location
  - Intersection checks are required to avoid front overlap

+ Surface grid preserved
+ Specialized layers near surfaces
- Computationally complex
- Low quality

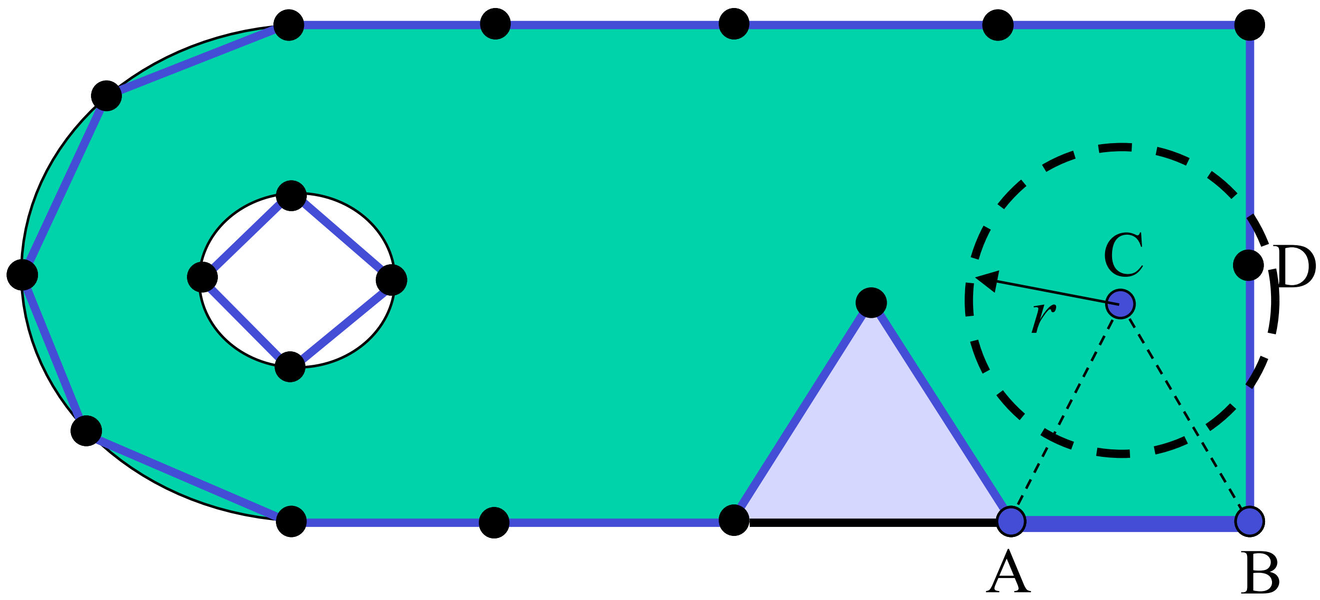


Advancing Front



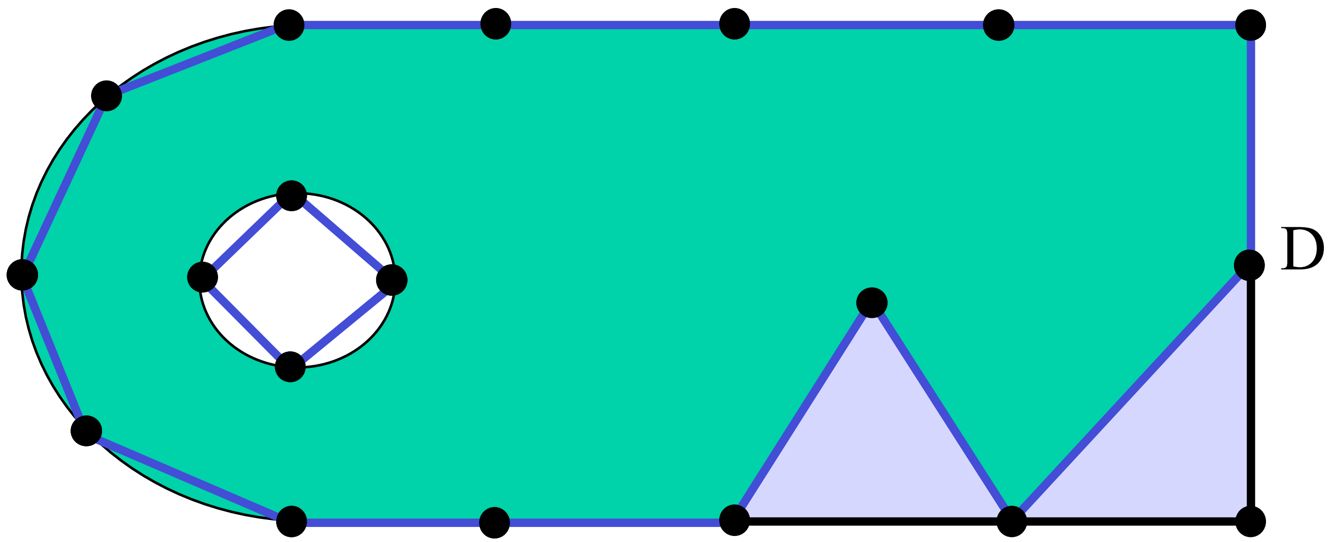
•Begin with boundary mesh - define as initial *front*
•For each edge (face) on front, locate ideal node C based on front AB

Advancing Front



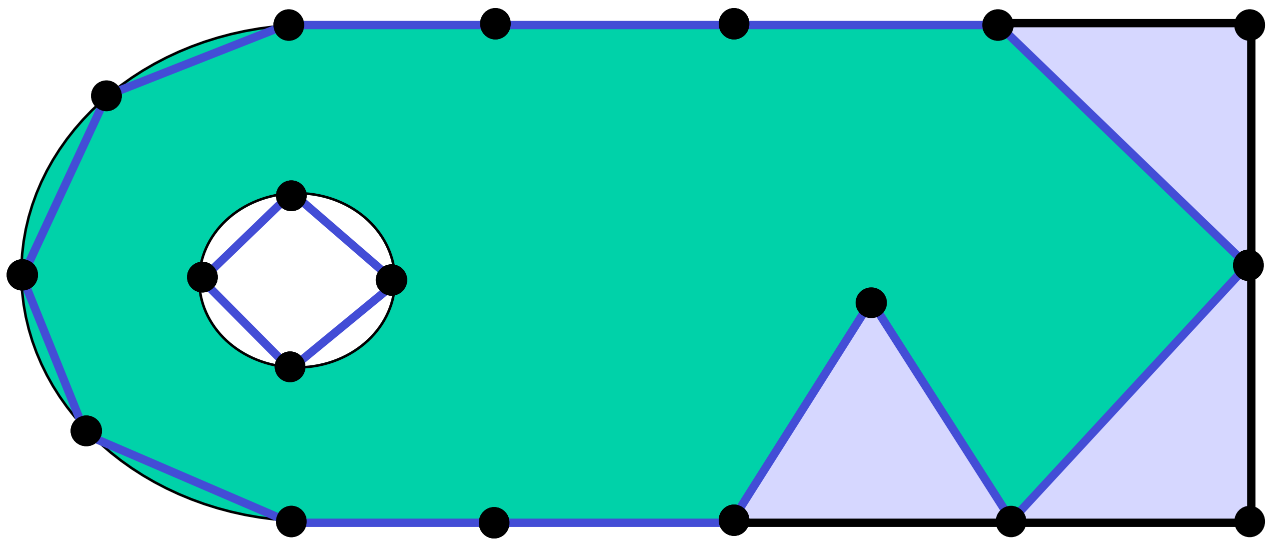
•Determine if any other nodes on current front are within search radius *r* of ideal location C (Choose D instead of C)

Advancing Front



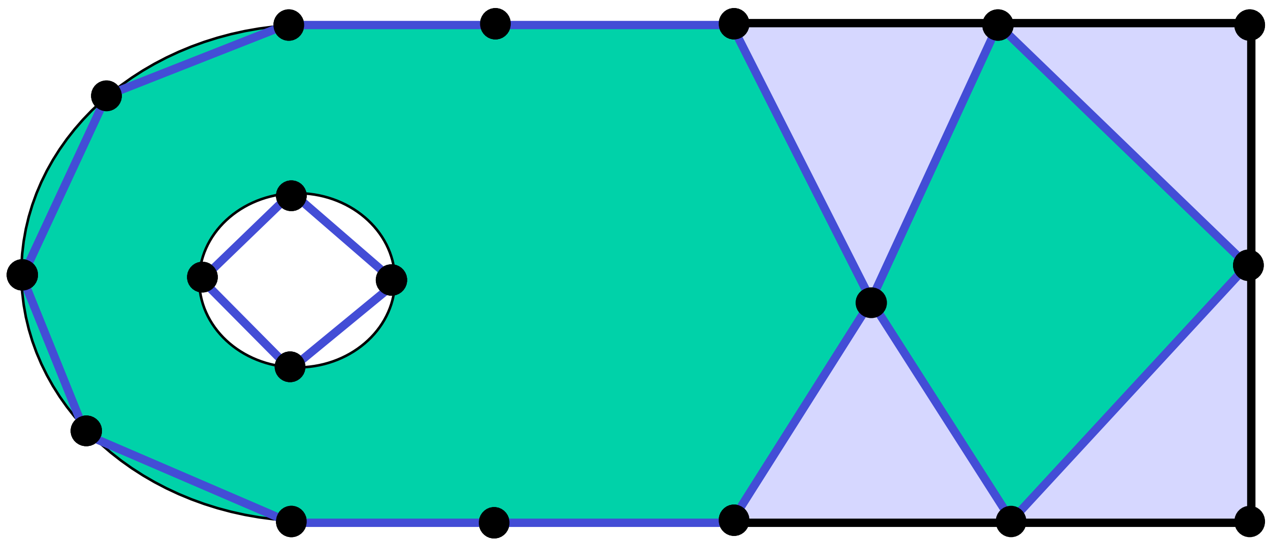
•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front



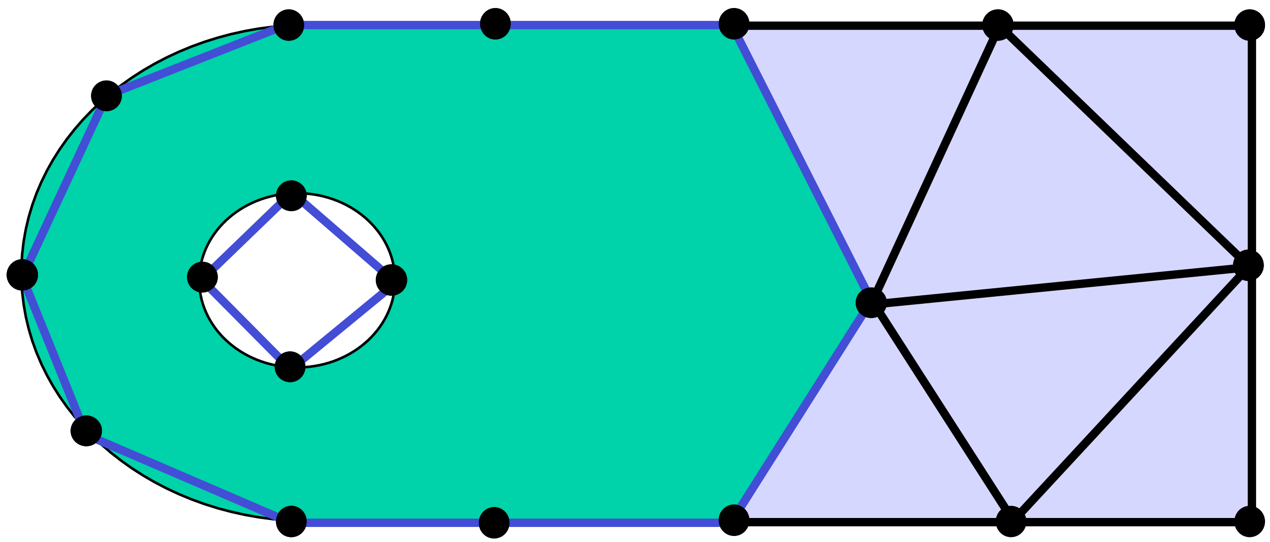
•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front



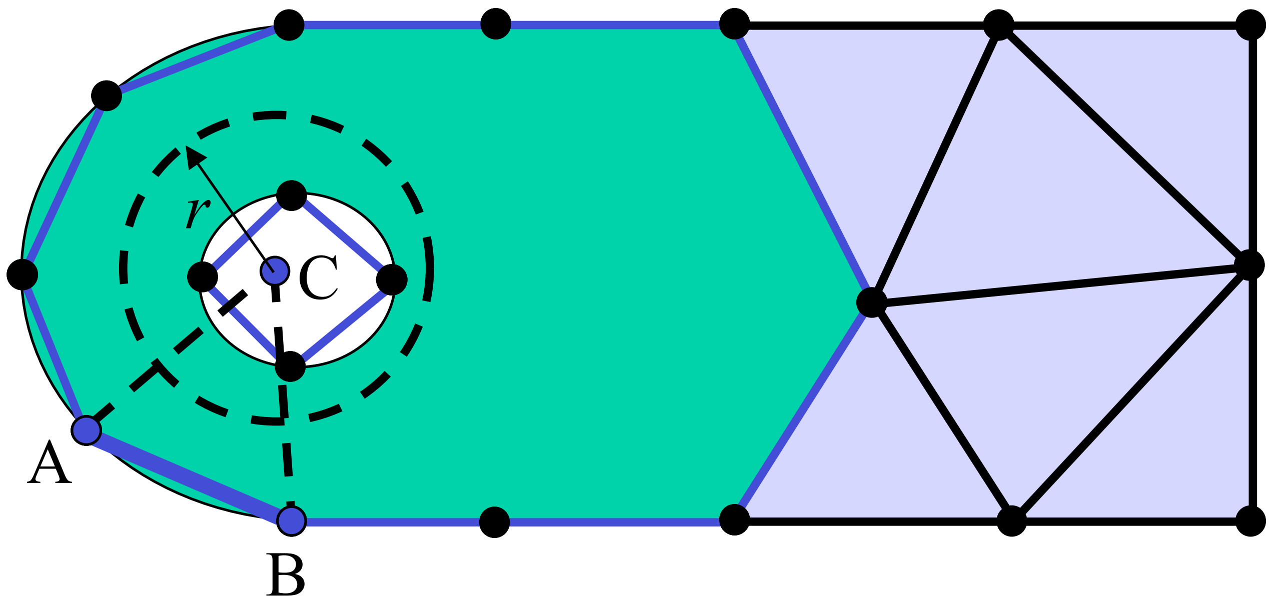
•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front



•Book-Keeping: New *front edges* added and deleted from *front* as triangles are formed
•Continue until no *front edges* remain on *front*

Advancing Front



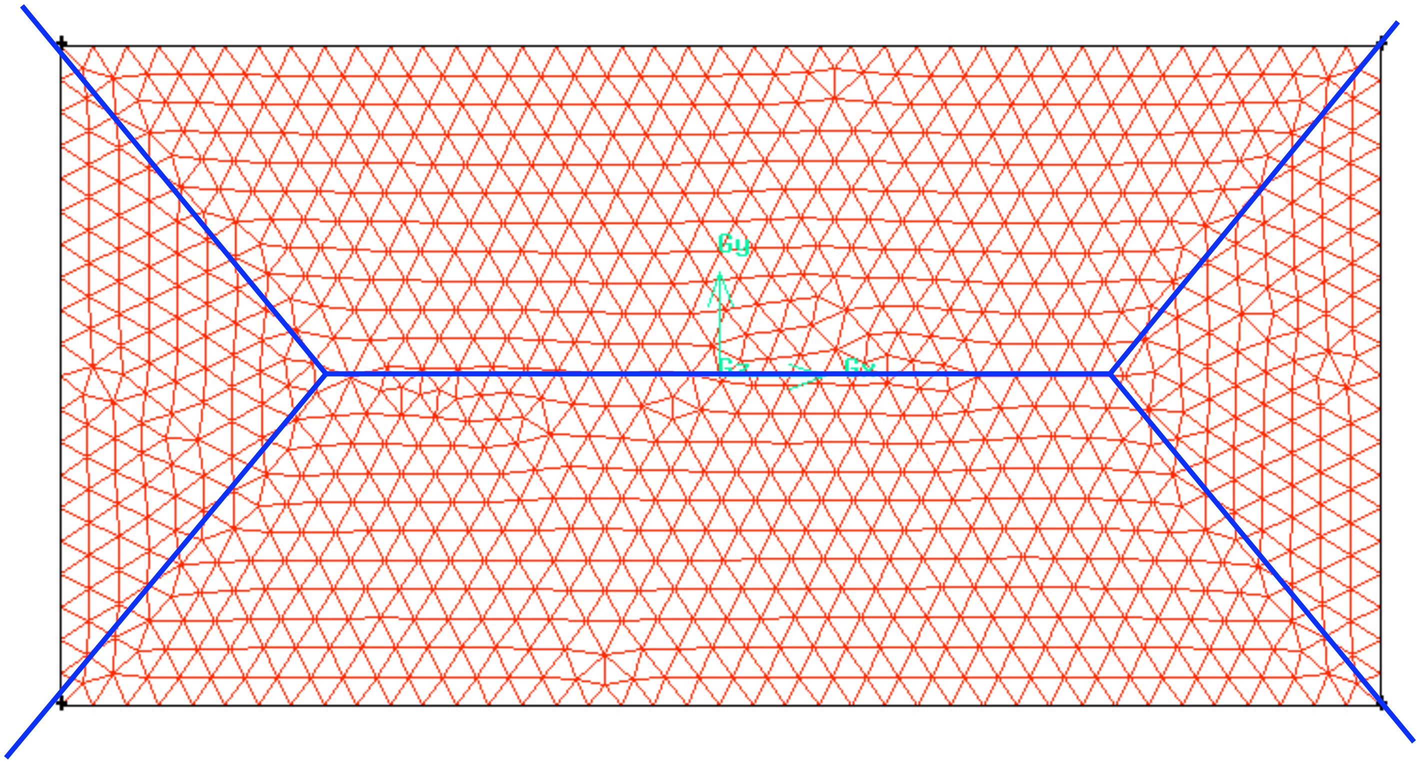
•Where multiple choices are available, use best quality (closest shape to equilateral)

•Reject any that would intersect existing front

•Reject any inverted triangles (|AB X AC| > 0)

•(Lohner,88;96)(Lo,91)

Advancing Front

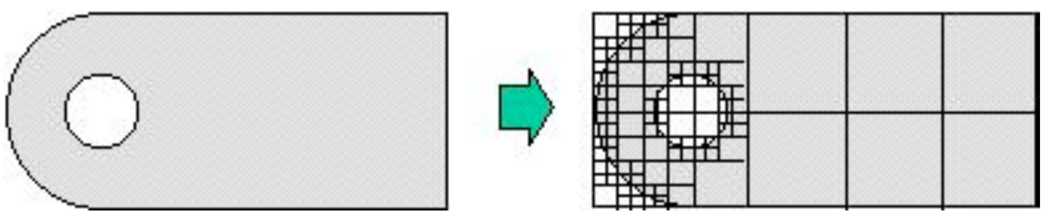


Remarkable high-quality grid

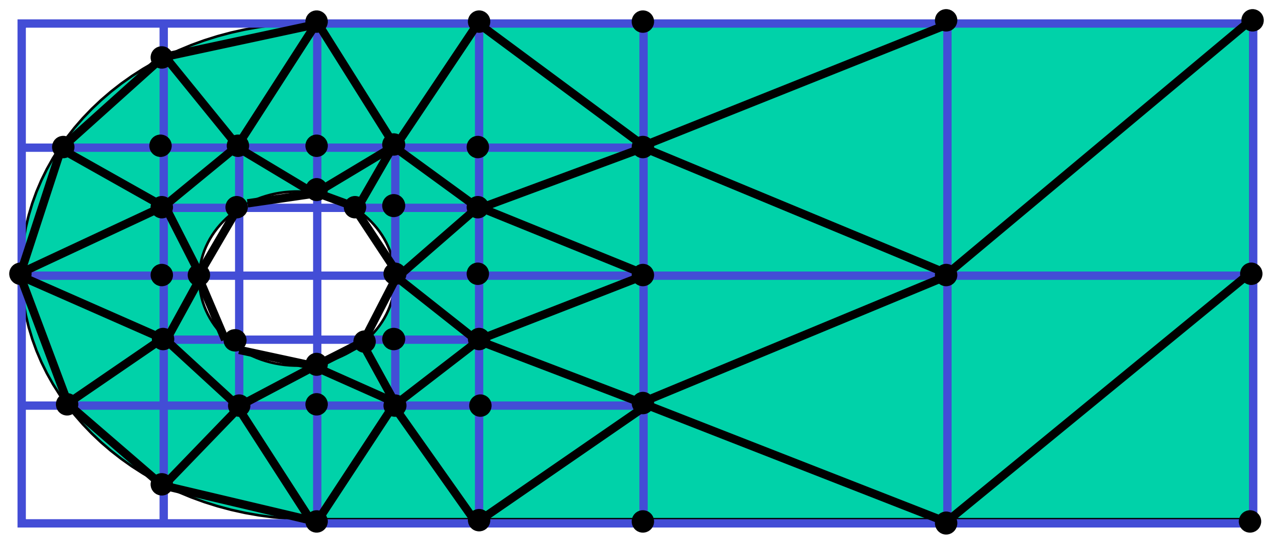
Unstructured Grids: Triangulations

- OCTREE
  - Squares containing the boundaries are recursively subdivided until desired resolution is obtained
  - Irregular cells (or triangulation) are generated near the surface where square intersect the boundary

+ Requires least of surface representation
+ Highly automated
- Cannot match surface grid
- Low quality near surfaces



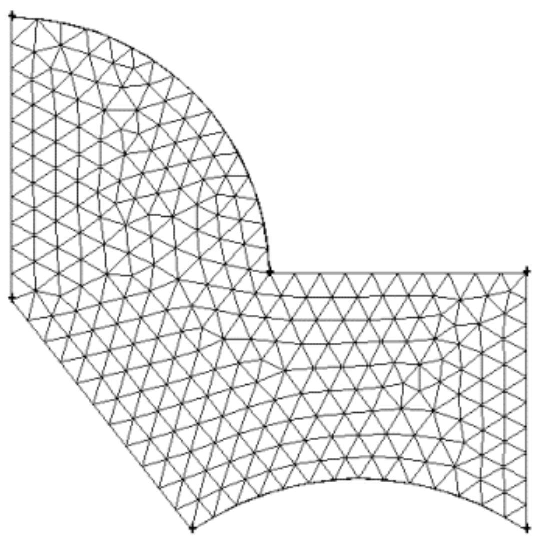
Octree/Quadtree



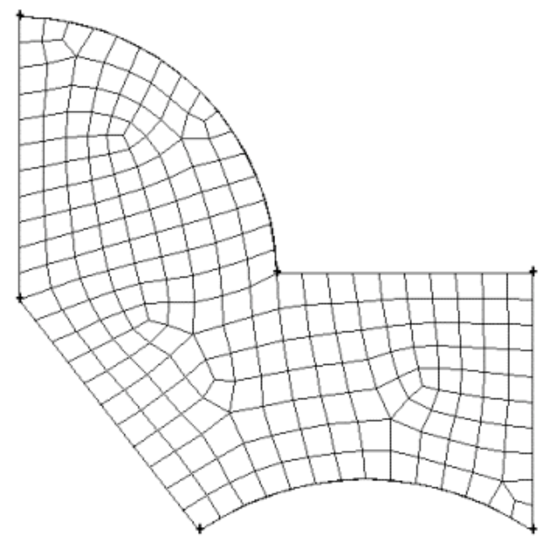
- •Define intial bounding box (*root* of quadtree)
- •Recursively break into 4 *leaves* per *root* to resolve geometry
- •Find intersections of leaves with geometry boundary
- •Mesh each *leaf* using corners, side nodes and intersections with geometry
- •Delete Outside
- •(Yerry and Shephard, 84), (Shepherd and Georges, 91)

Unstructured Grids: Paving

- Advancing front technique based on quads (instead of triangles)
- Only in 2D

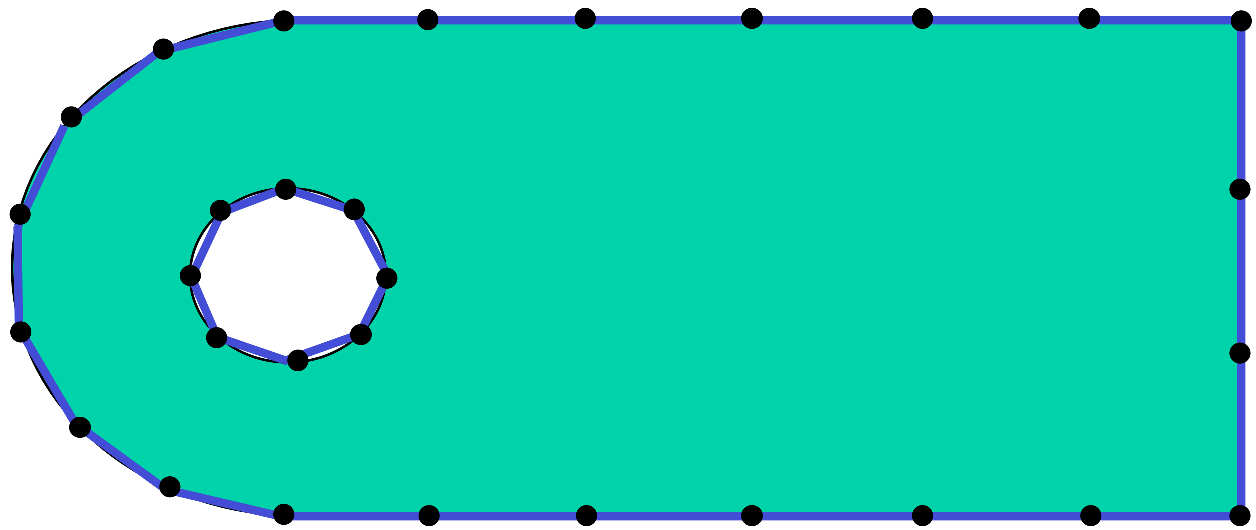


Triangulation



Paving

Unstructured-Quad

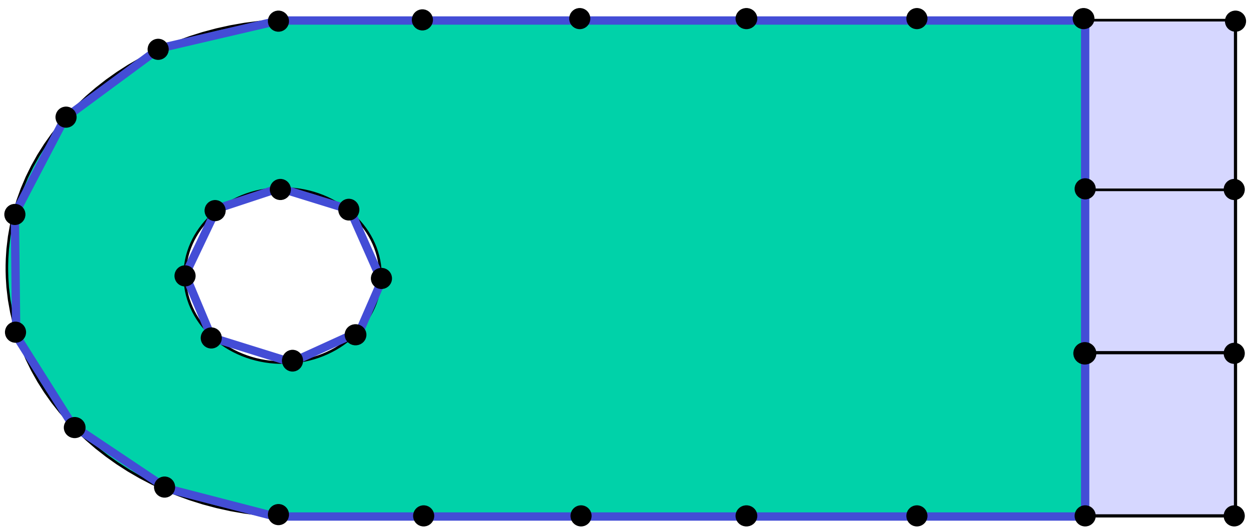


Paving

•Advancing Front: Begins with front at boundary
•Forms rows of elements based on front angles
•Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

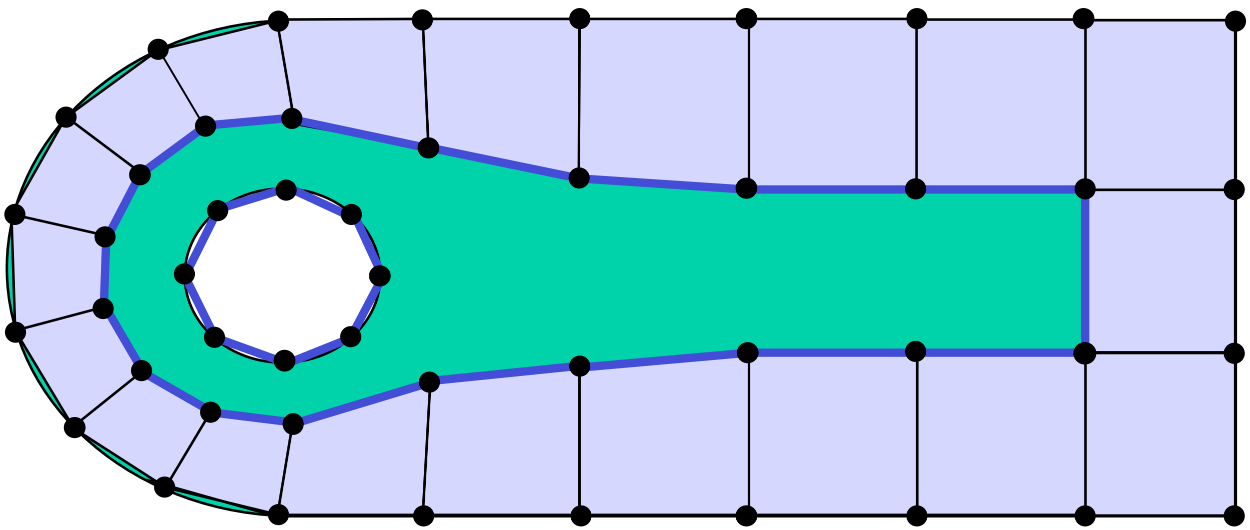


Paving

•Advancing Front: Begins with front at boundary
•Forms rows of elements based on front angles
•Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad



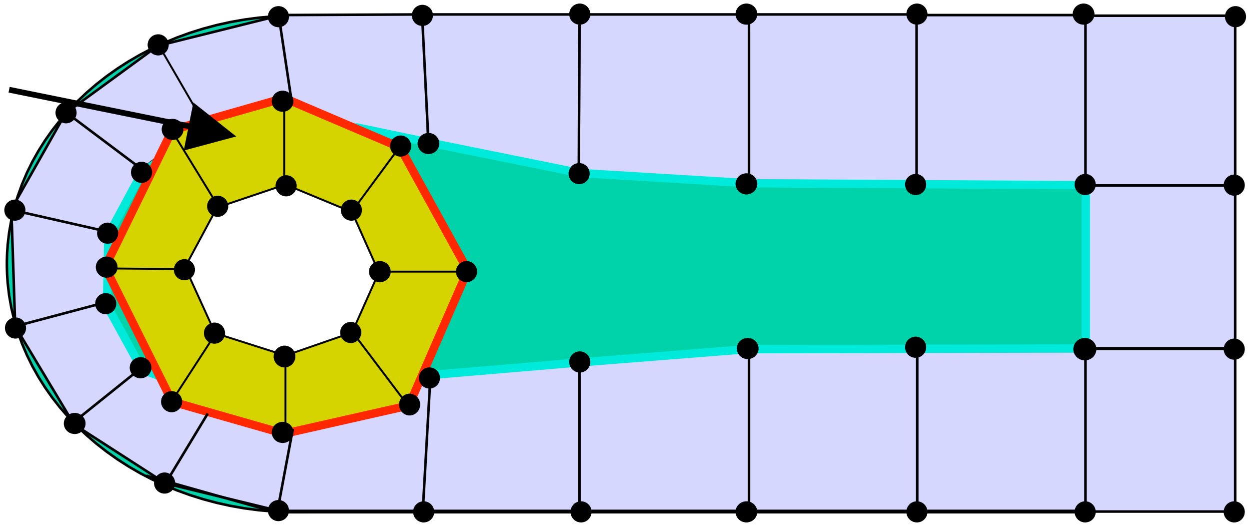
Paving

•Advancing Front: Begins with front at boundary
•Forms rows of elements based on front angles
•Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

Form new row and check for overlap

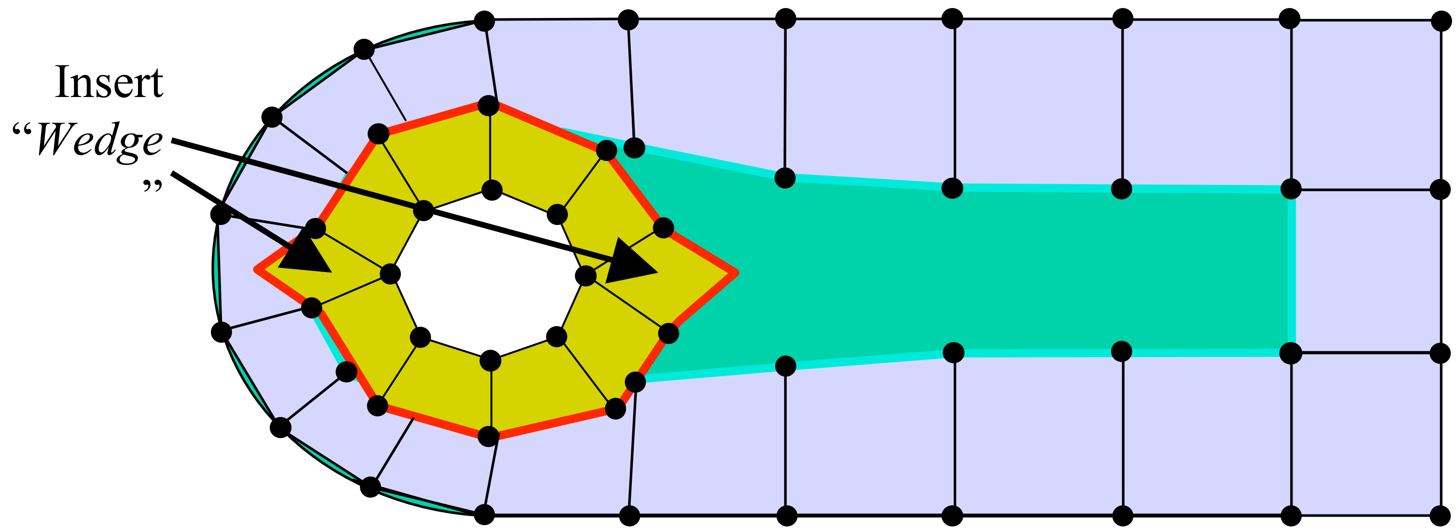


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

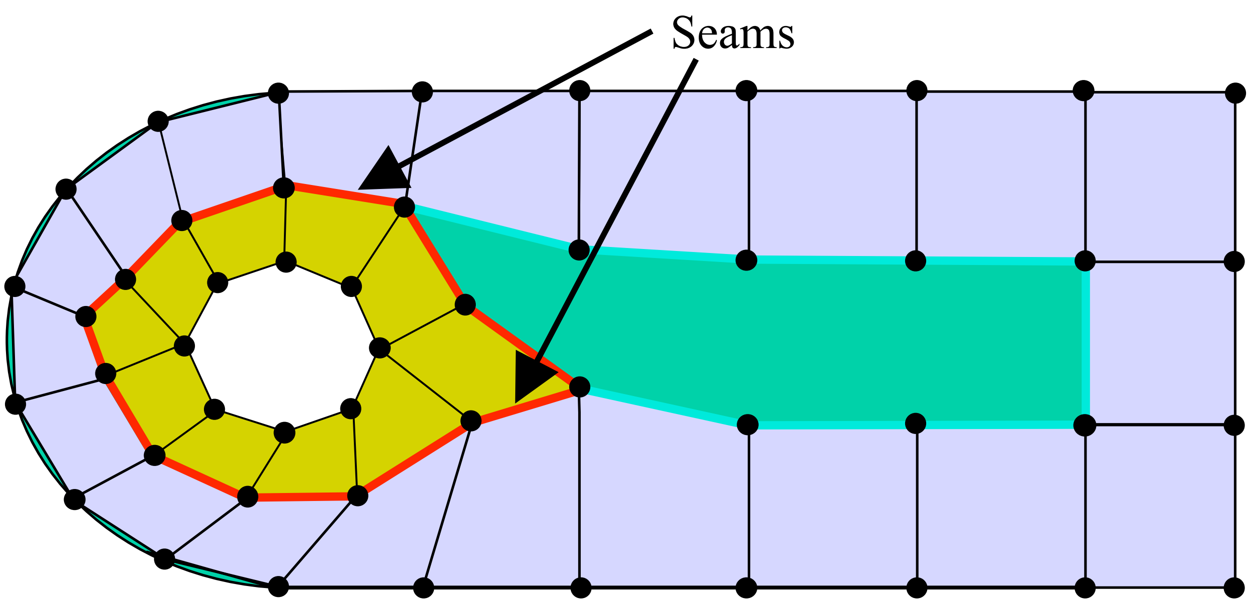


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad



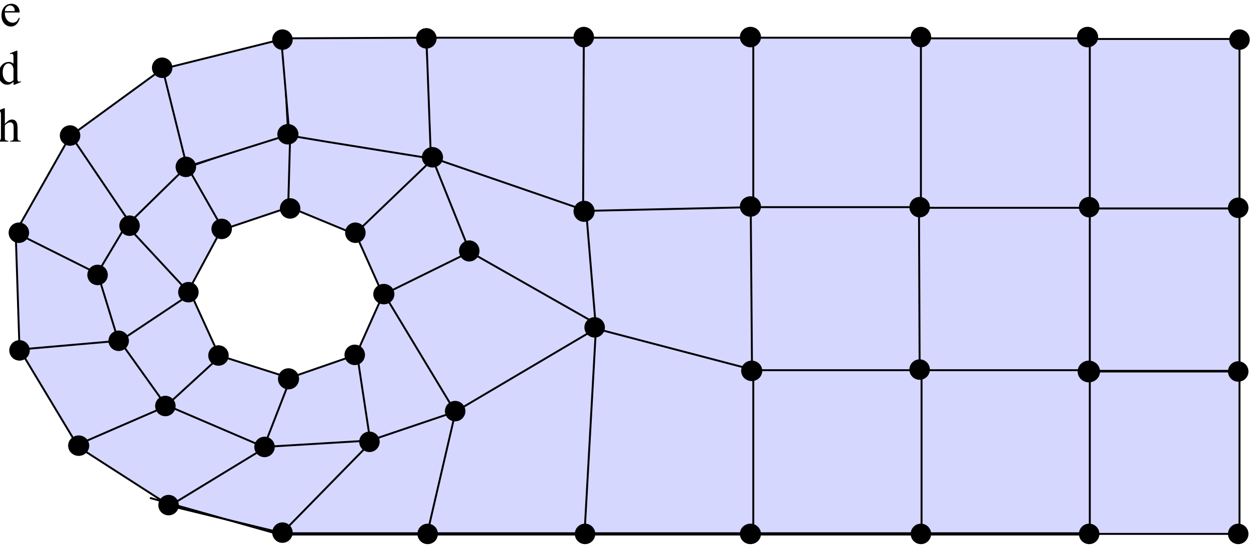
Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

(Blacker,92)(Cass,96)

Unstructured-Quad

Close
Loops and
smooth

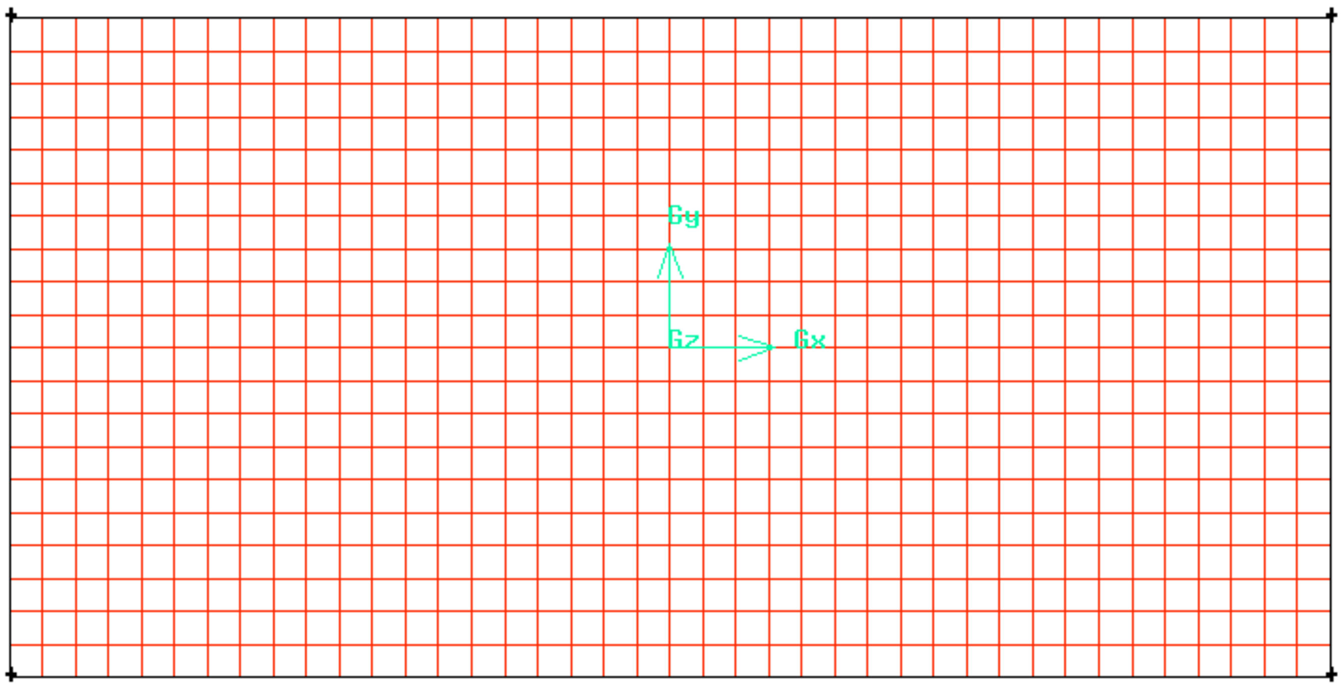


Paving

- Advancing Front: Begins with front at boundary
- Forms rows of elements based on front angles
- Must have even number of intervals for all-quad mesh

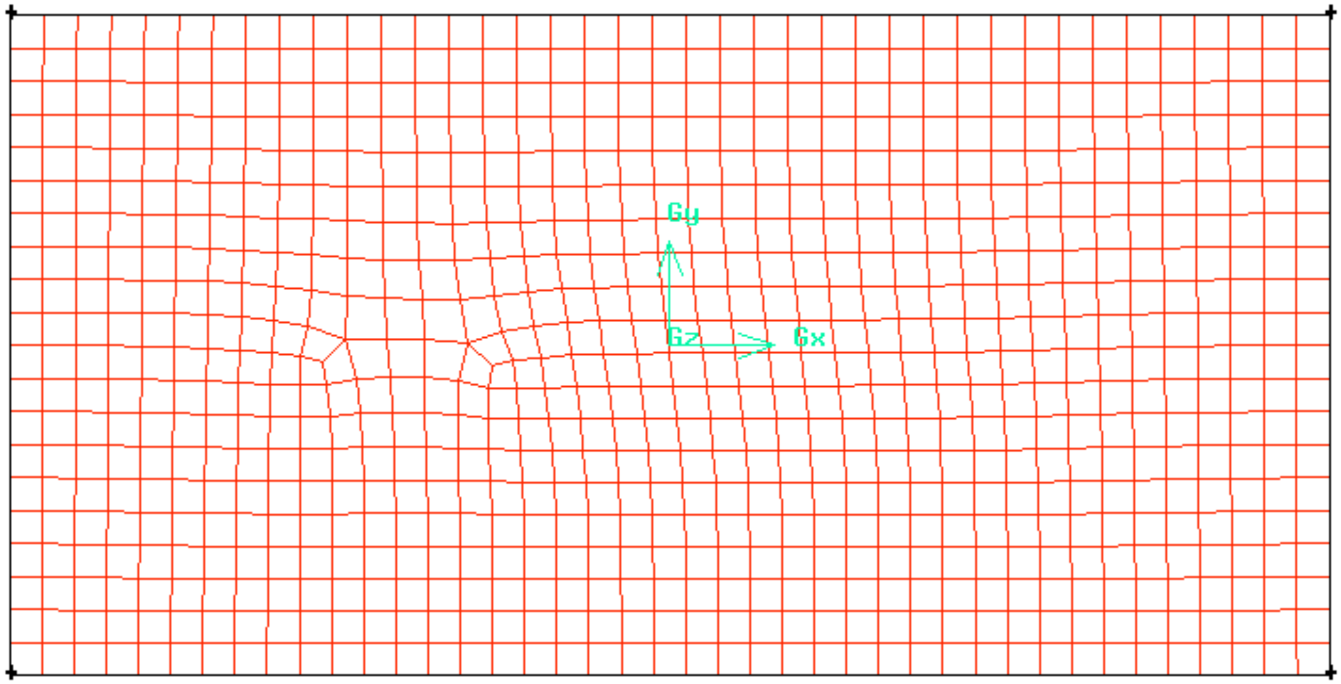
(Blacker,92)(Cass,96)

Unstructured-Quad



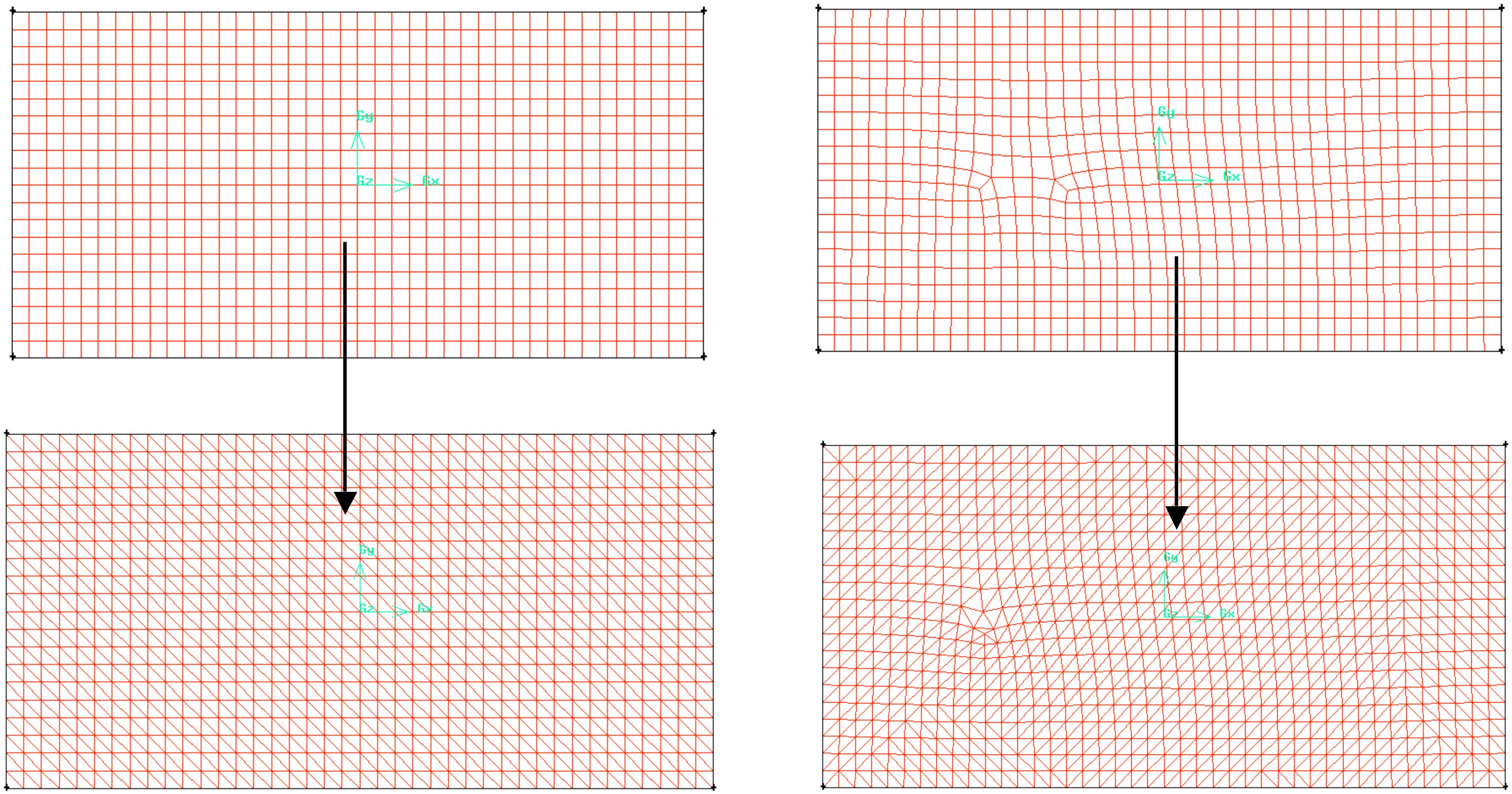
Reproduces an uniform mesh

Unstructured-Quad

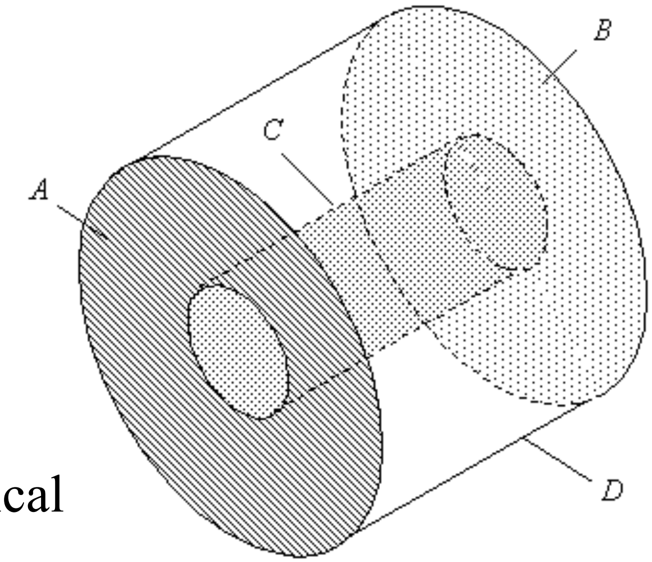


Reproduces an uniform mesh…almost. But it allows flexibility in the edge meshing

Unstructured Quad-to-Tri



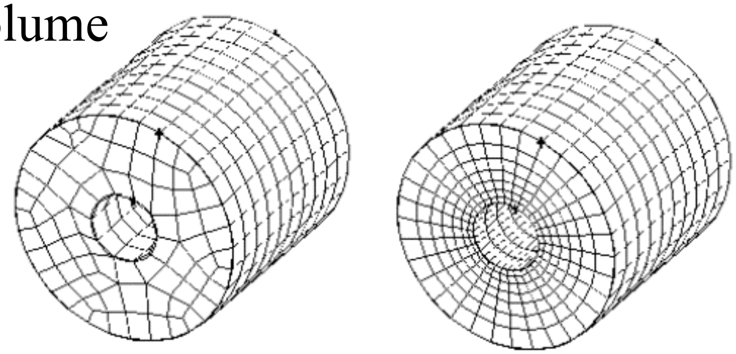
Unstructured Grids: Coopering



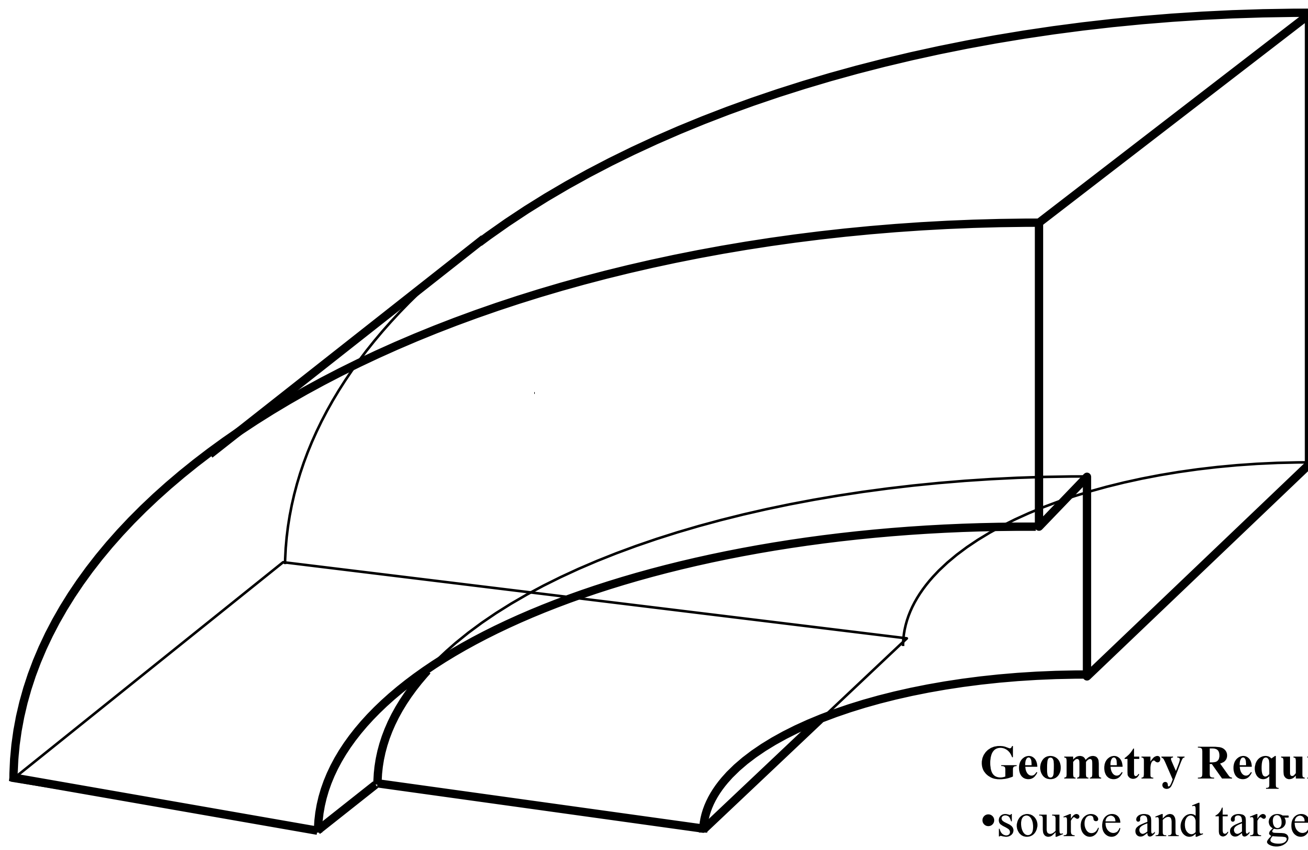
- 2D mesh sweeping
  - Only for cylindrical volumes
  - unstructured surface mesh is generated on surface A (source face)

  - structured grids are generated on cylindrical surfaces C & D

  - mesh on surface A is sweeped in the volume to generate the full 3D m esh



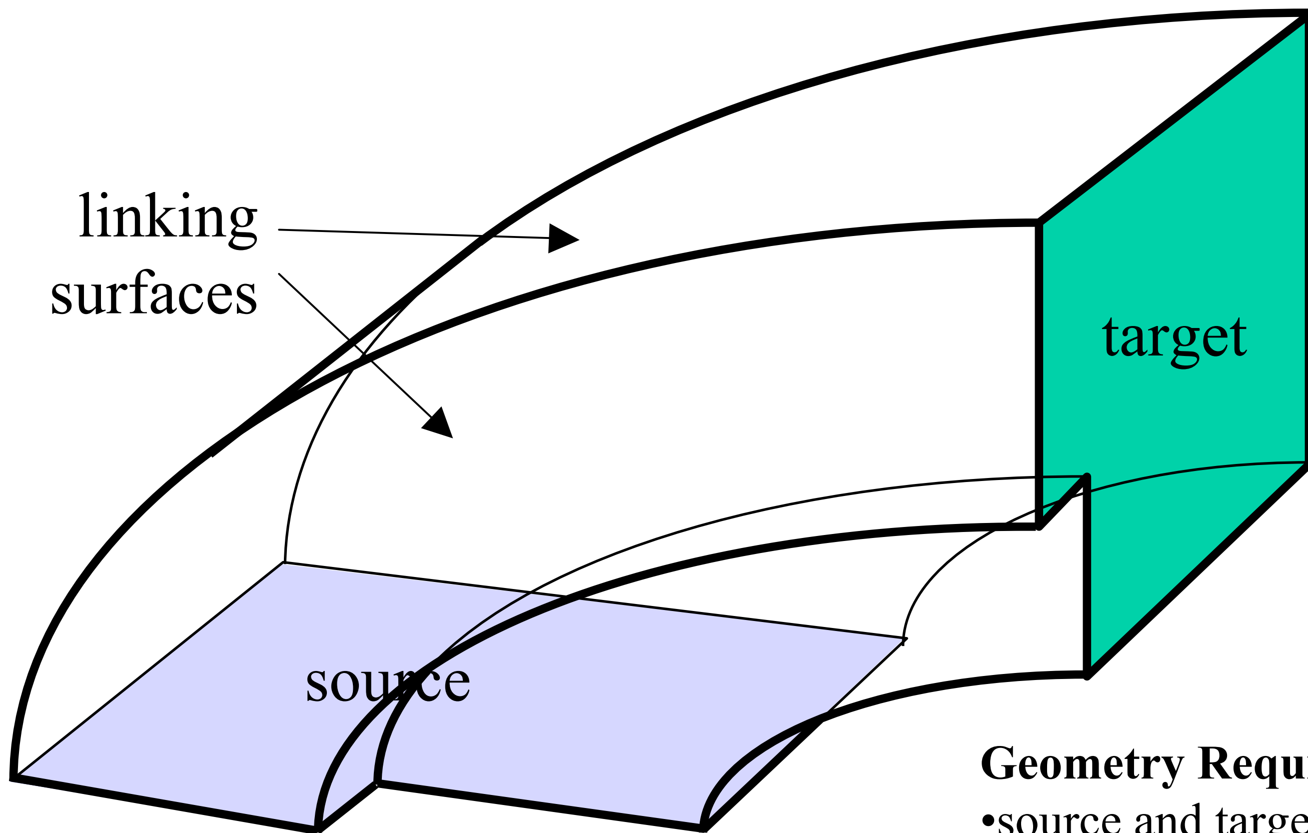
Coopering/Sweeping



Sweeping

Geometry Requirements
•source and target surfaces topologicaly similar
•linking surfaces *mapable* or *submapable*

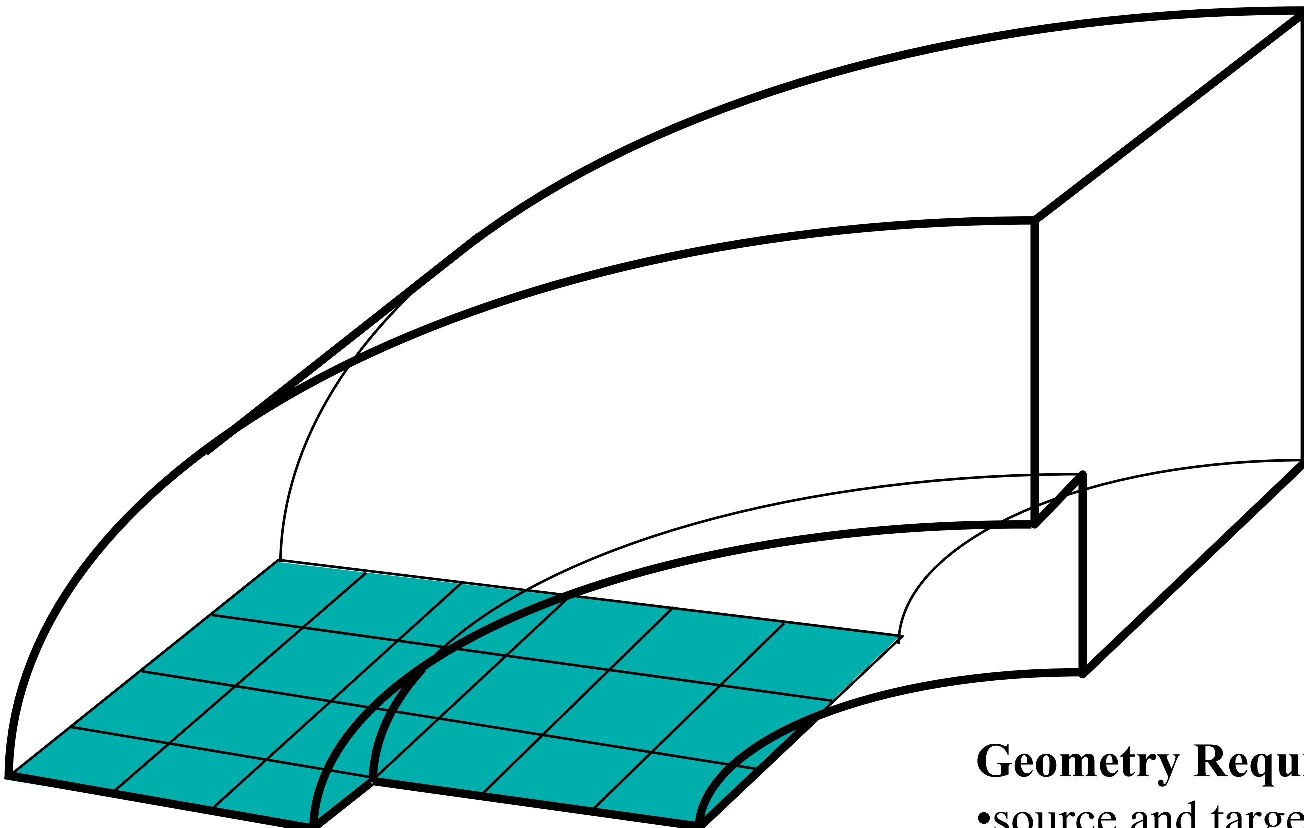
Coopering/Sweeping



Sweeping

Geometry Requirements
•source and target surfaces topologicaly similar
•linking surfaces *mapable* or *submapable*

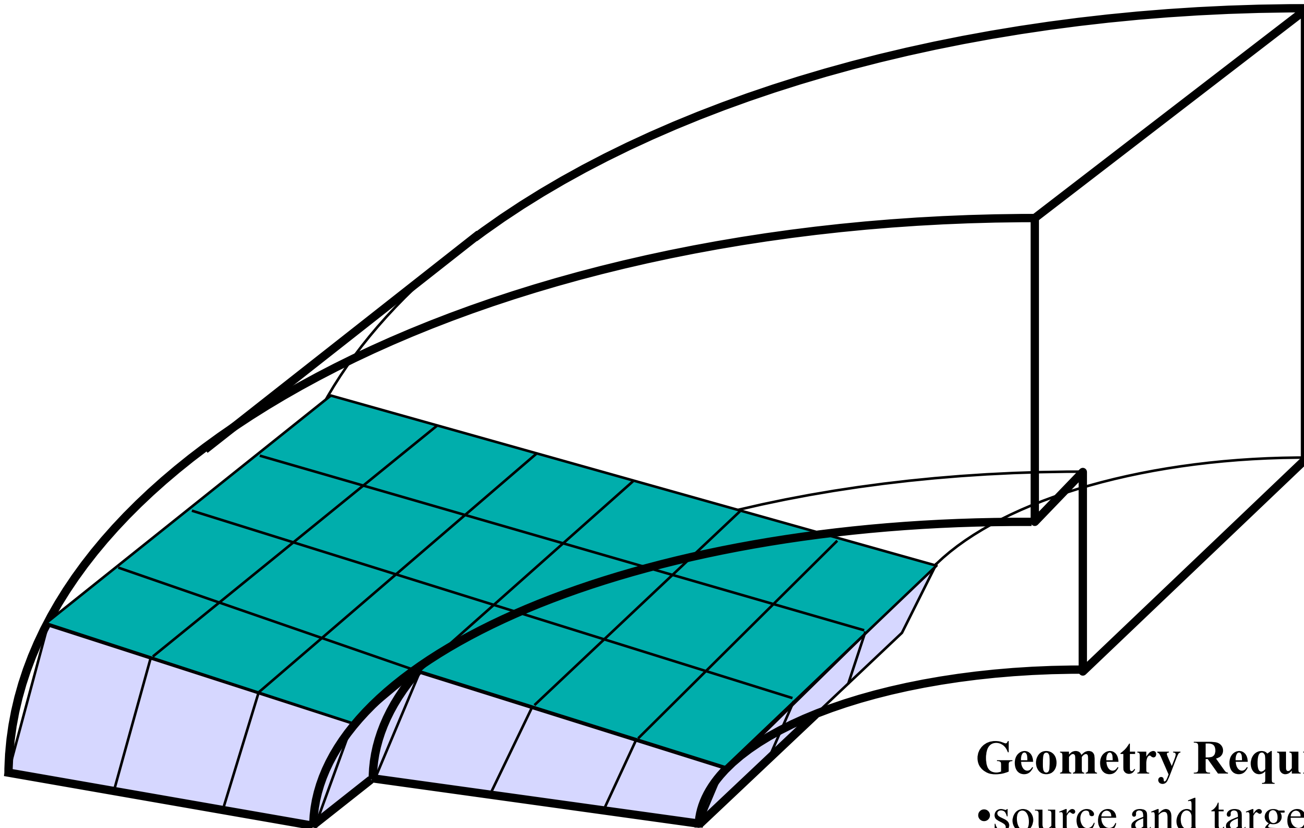
Coopering/Sweeping



Sweeping

Geometry Requirements
•source and target surfaces topologicaly similar
•linking surfaces *mapable* or *submapable*

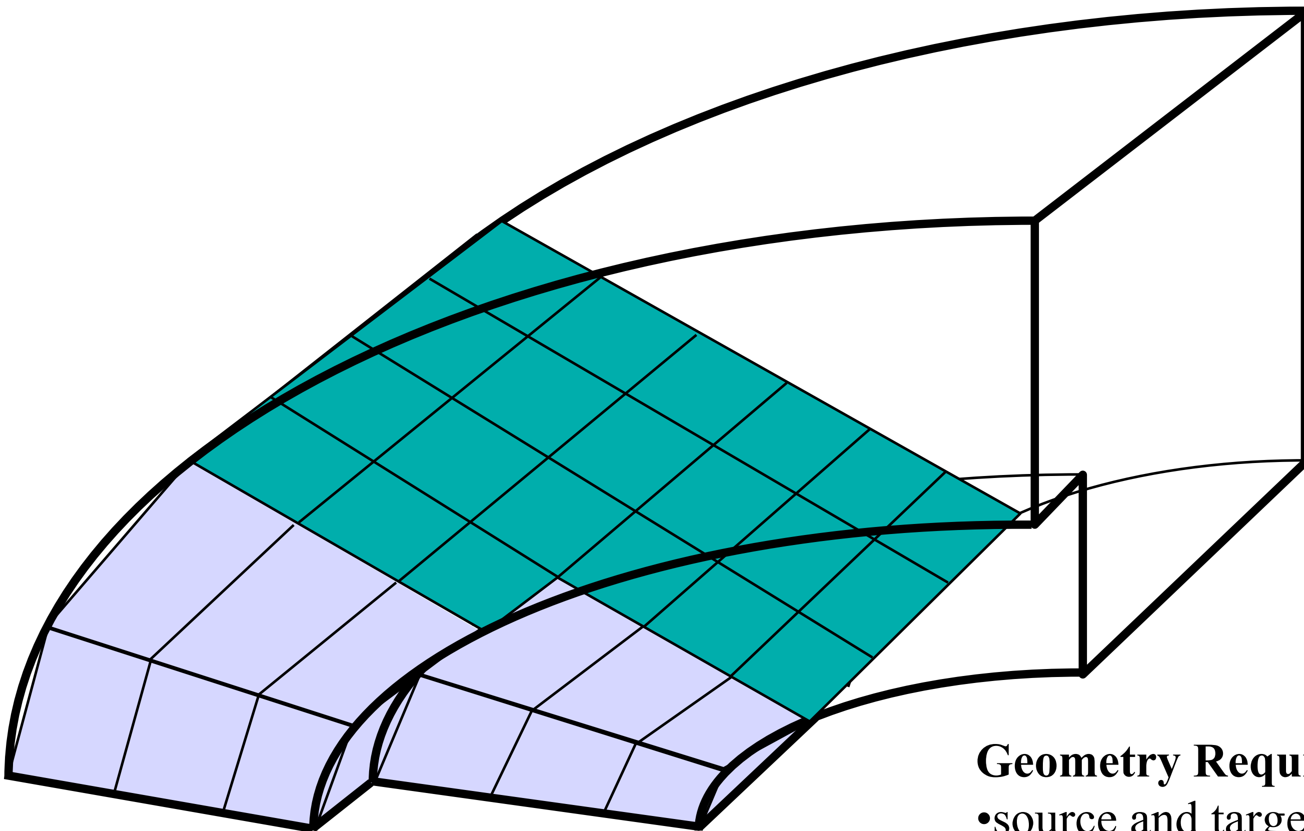
Coopering/Sweeping



Sweeping

Geometry Requirements
•source and target surfaces topologicaly similar
•linking surfaces *mapable* or *submapable*

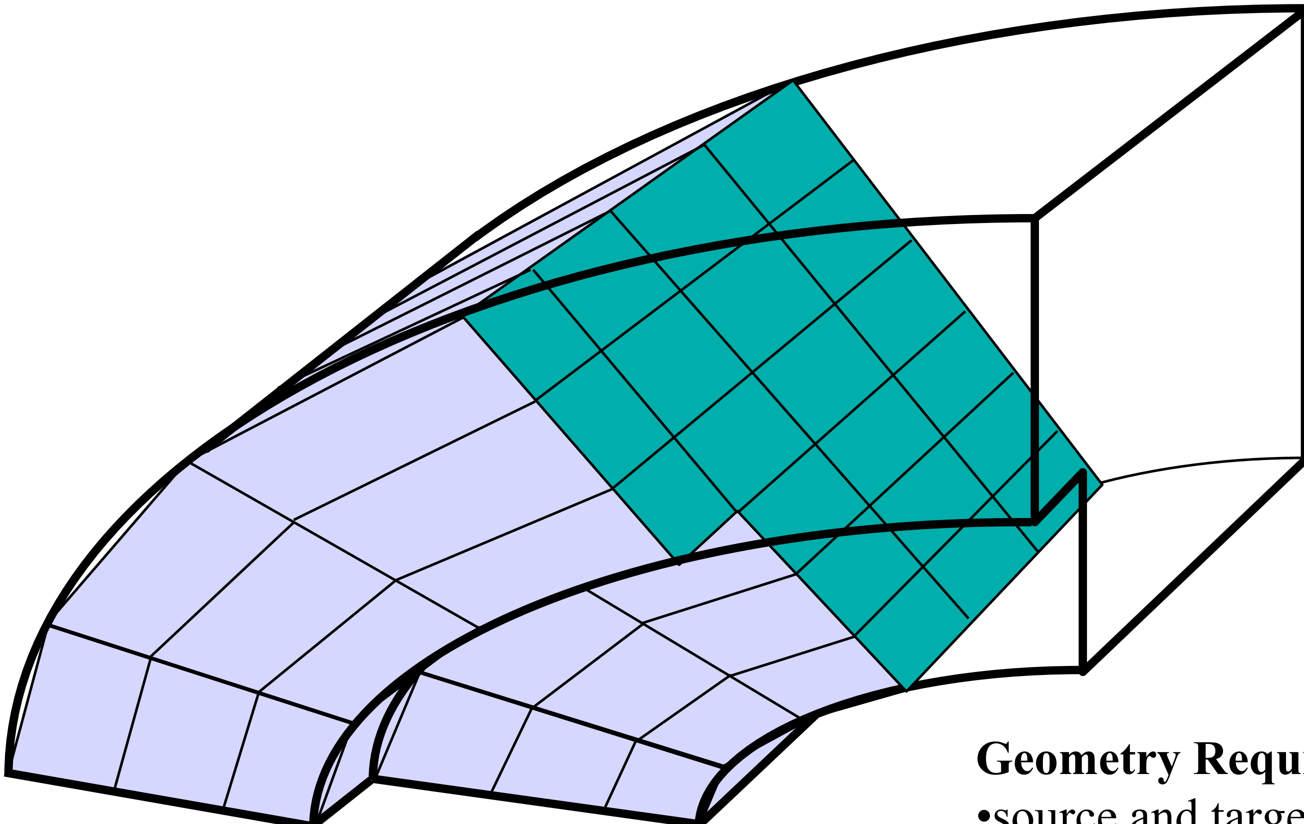
Coopering/Sweeping



Geometry Requirements
•source and target surfaces topologicaly similar
•linking surfaces *mapable* or *submapable*

Sweeping

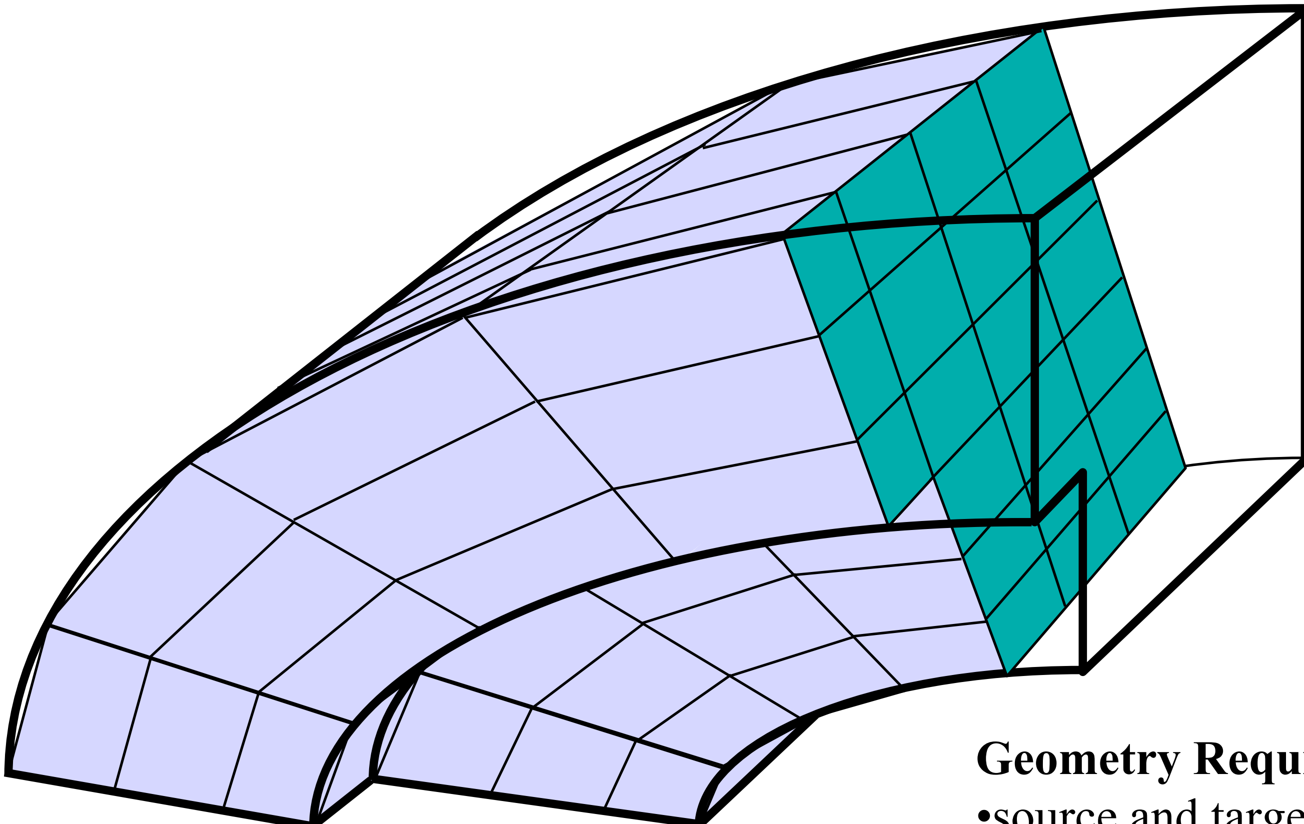
Coopering/Sweeping



Sweeping

Geometry Requirements
•source and target surfaces topologicaly similar
•linking surfaces *mapable* or *submapable*

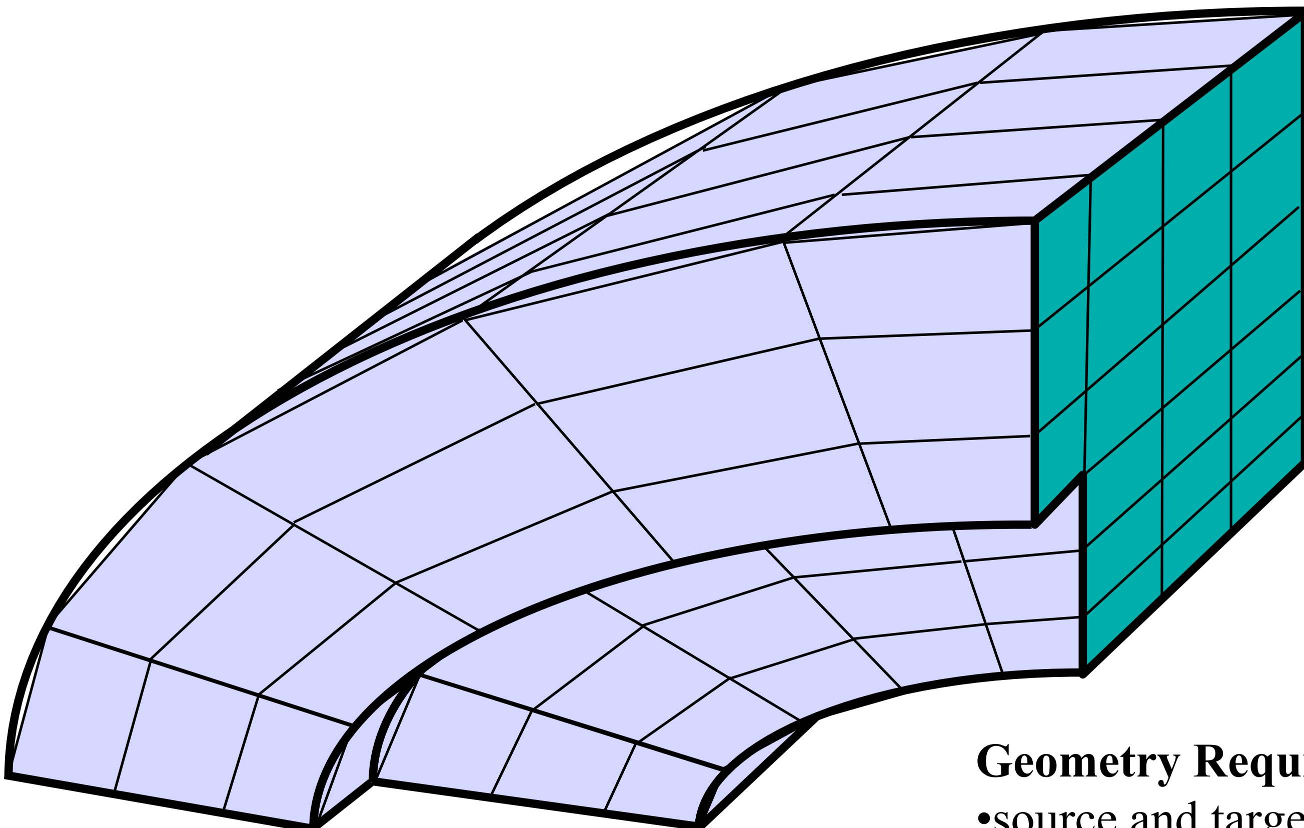
Coopering/Sweeping



Sweeping

Geometry Requirements
•source and target surfaces topologicaly similar
•linking surfaces *mapable* or *submapable*

Coopering/Sweeping



Sweeping

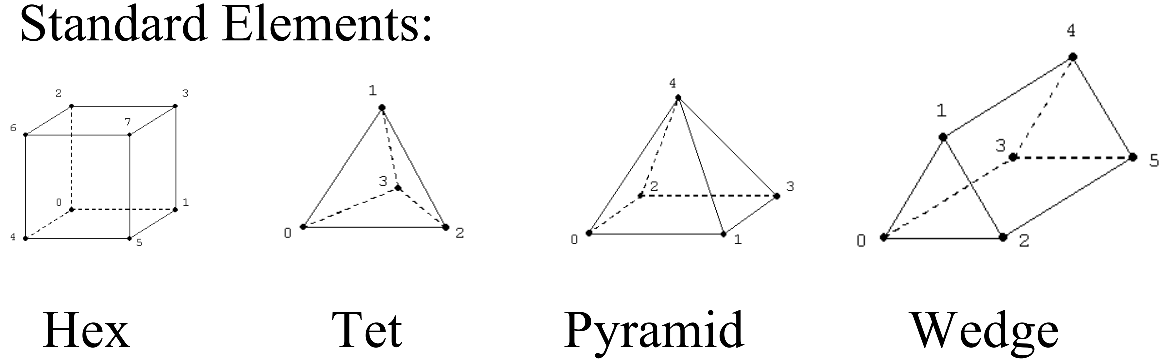
Geometry Requirements

•source and target surfaces topologicaly similar

•linking surfaces *mapable* or *submapable*

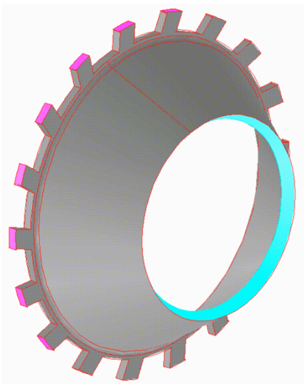
Unstructured Grids: 3D elements

- Standard Elements:



- Hex: Maximum Volume Covered per Edge Size
- Hex: Maximum Ratio Nodes/Elements
- Hex/Wedges: Clustering at Solid Wall with High Quality Elements
- Tets: Automatic Meshing of Extremely Complicated Regions
- Pyramids/Wedges: Transition Between Tets & Hex

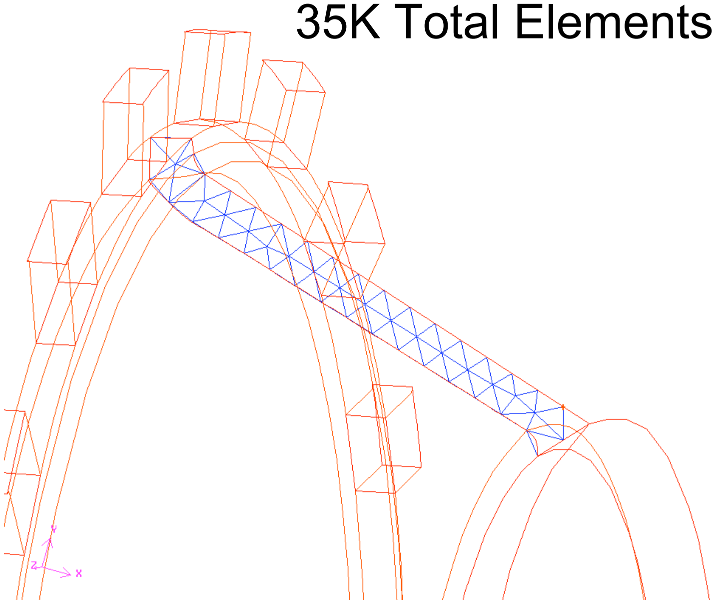
Unstructured Grids: Hex or Tets?



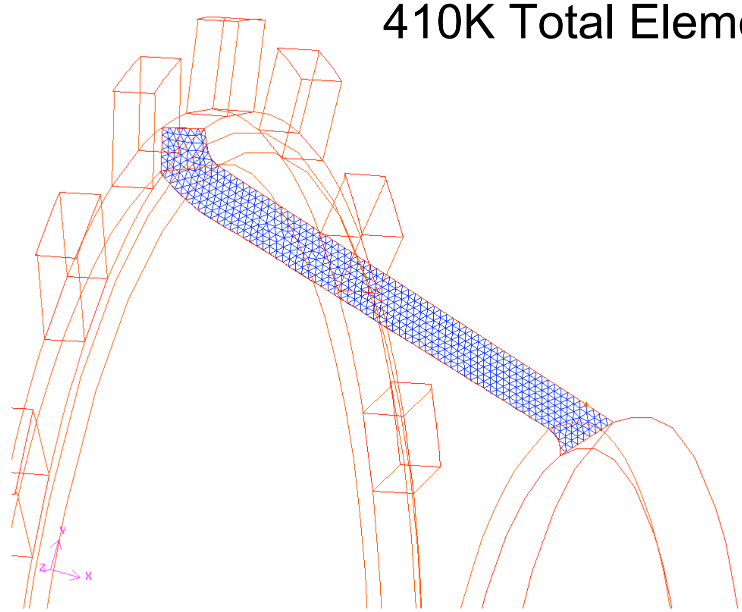
We NEED Hex-Based Meshing because:

- Equiangular Tets are NOT Good for Thin Volumes
- Too Many Elements for Reasonable Resolutions
  (estimated >2M grid Points in conical-annular Swirler)

35K Total Elements



410K Total Elements



Cross-Section of the Swirler

What is available in GAMBIT

	Speed	Robustness	Quality & Control	Complex Geometry	Mesh sizes
• Structured gridding (mapping)	+	+	+	-	+
• Unstructured triangulation (2D/3D)	-	+	+/-	+	-
• Unstructured paving (2D)	-	-	+/-	+	+
• Unstructured coopering (3D)	+	-	+/-	-	+

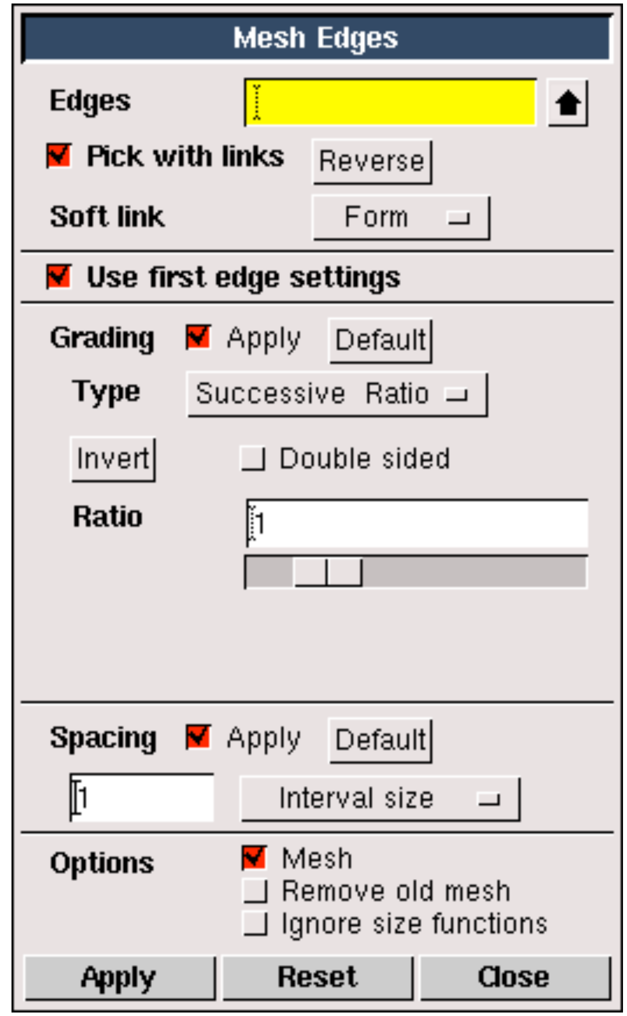
All GAMBIT meshes are exported as unstructured collection of (mixed) elements

Grid generation – 1D - Edges

Straightforward

Select number of points
Select distribution of points

Edge direction is defined from 1st to 2nd vertex
Clustering toward one side is defined accordingly



Grid generation – 2D - Faces

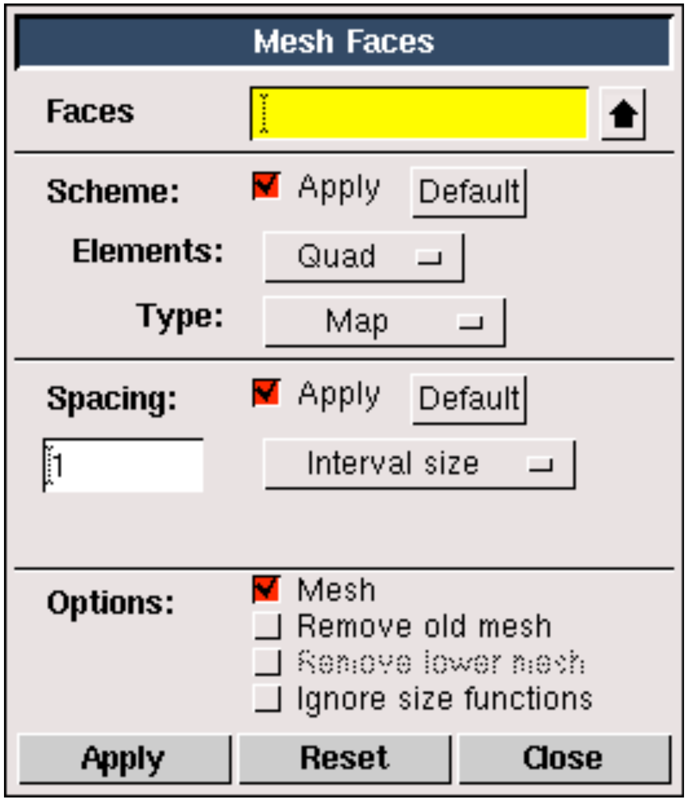
Easy

Select number of points

- use predefined edge meshes
- use uniform spacing

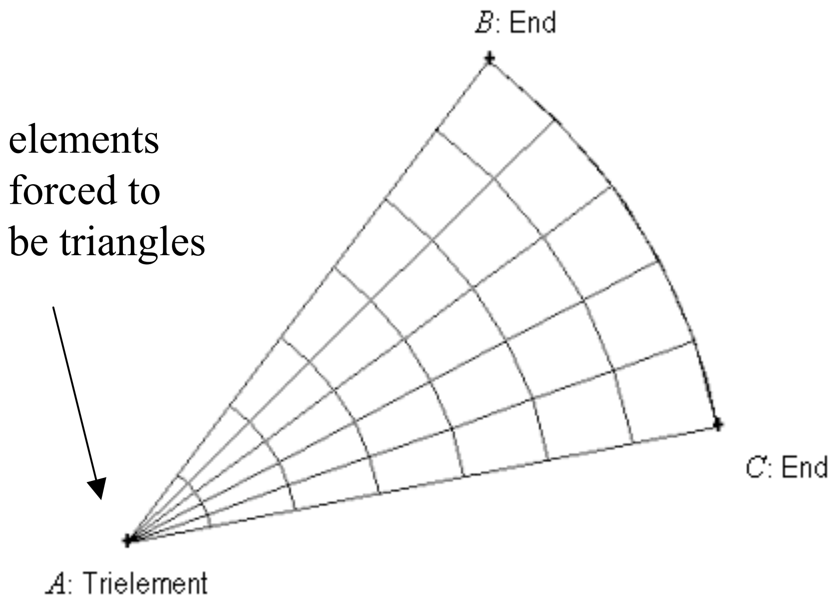
Select meshing scheme

- constraints on the edge meshing for mapping and paving schemes



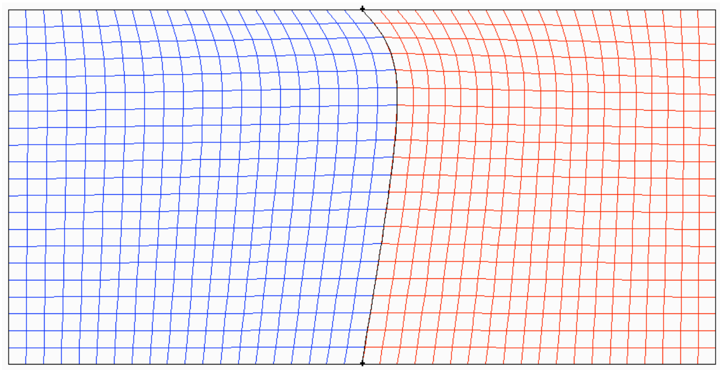
Grid generation – 2D - Faces

It is possible to force the cell element type at face-vertices

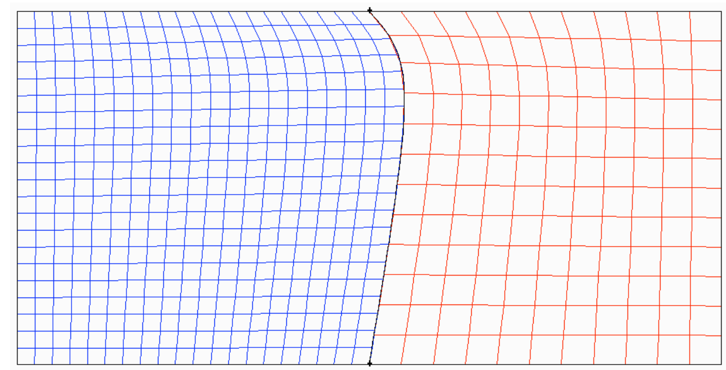


Mixing element-type is one of the main advantages of unstructured mesh technology

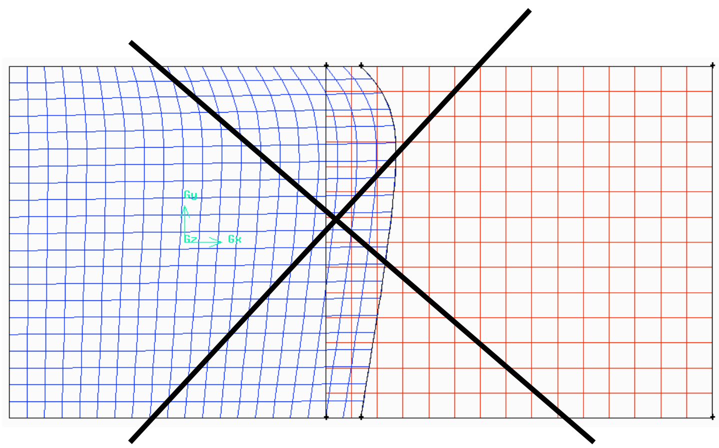
Grid generation – 2D - Mesh-patching options



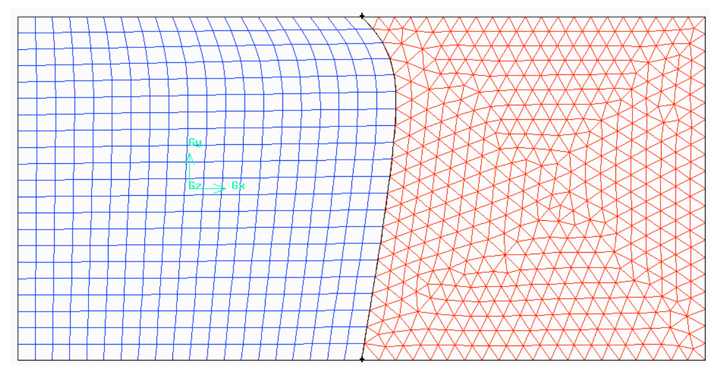
Matching interface



Non-conformal interface



Overlapping interface



Mixed-element interface

Grid generation – 3D - Volumes

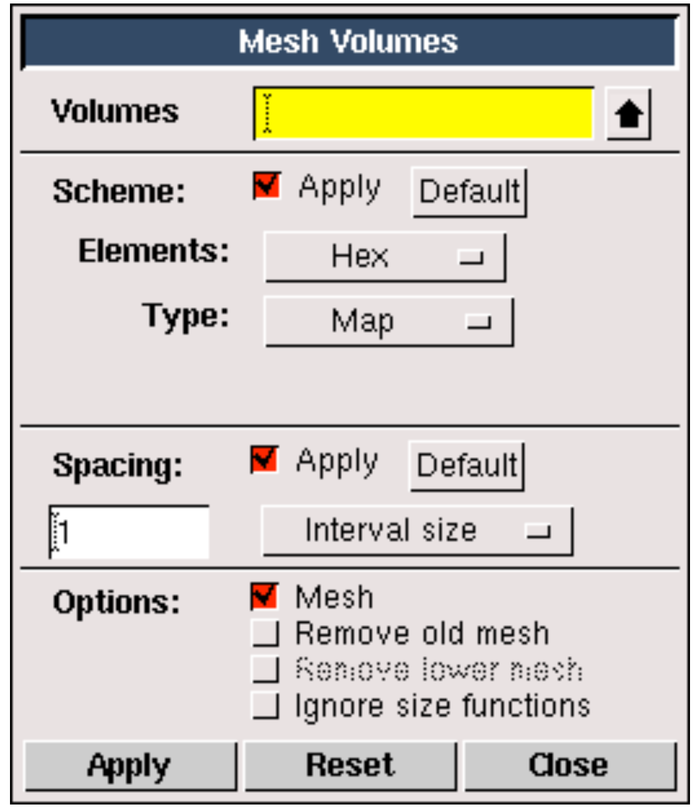
Not so easy

Select number of points

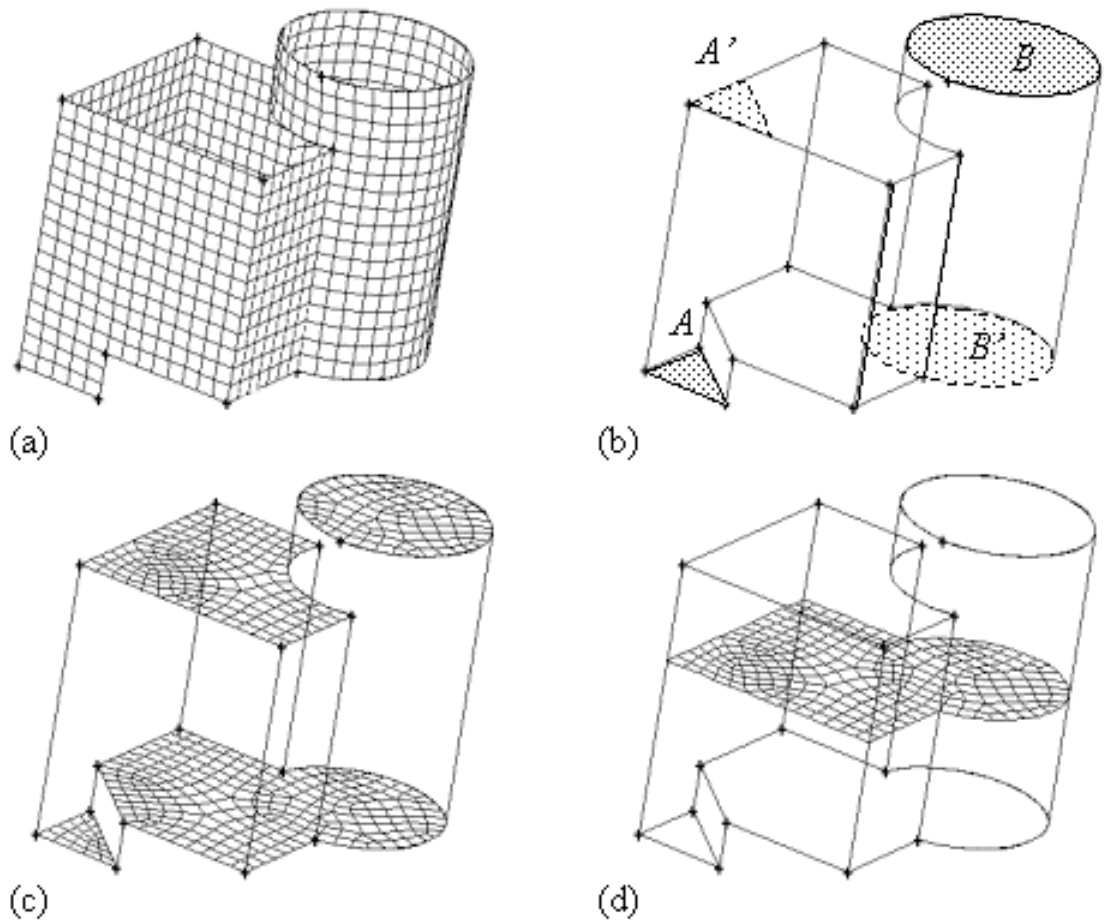
- use predefined face meshes
- use uniform spacing

Select meshing scheme

- constraints on the face meshing for mapping and cooper schemes



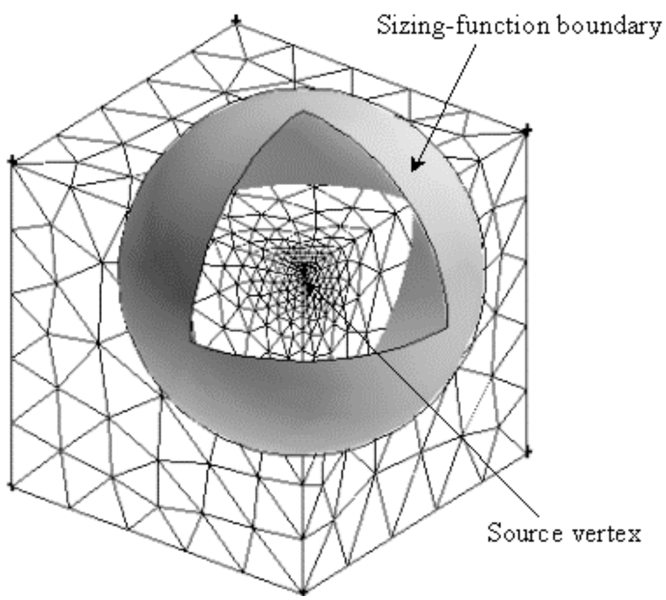
3D Grid generation – Advanced Cooper technique

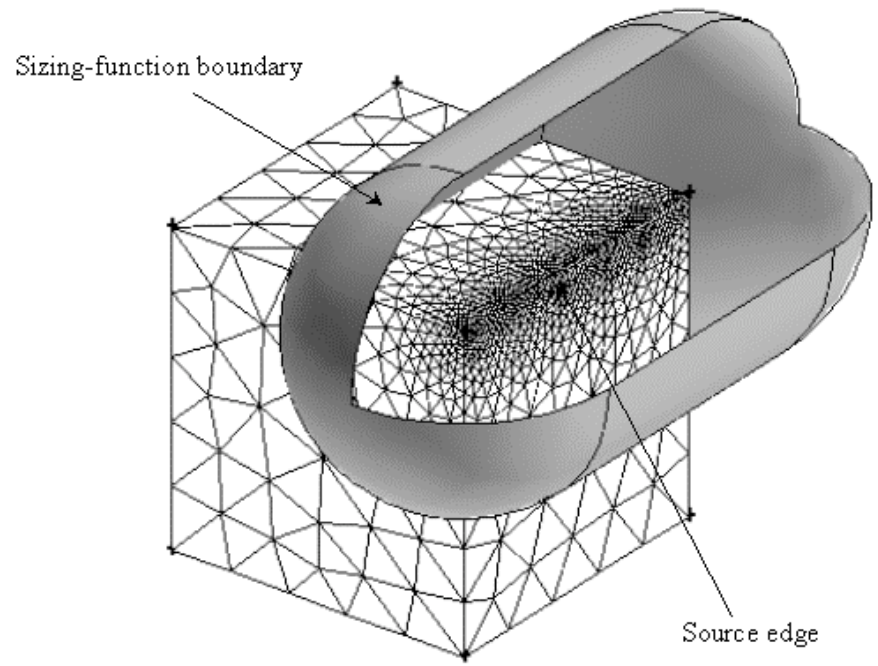


"Creative" way of coopering: multisurface to multisurface sweep

Grid generation – Sizing functions

Instead of the bottom-up approach (1D to 3D) grid generation
Sizing functions can be specified to mesh volumes directly

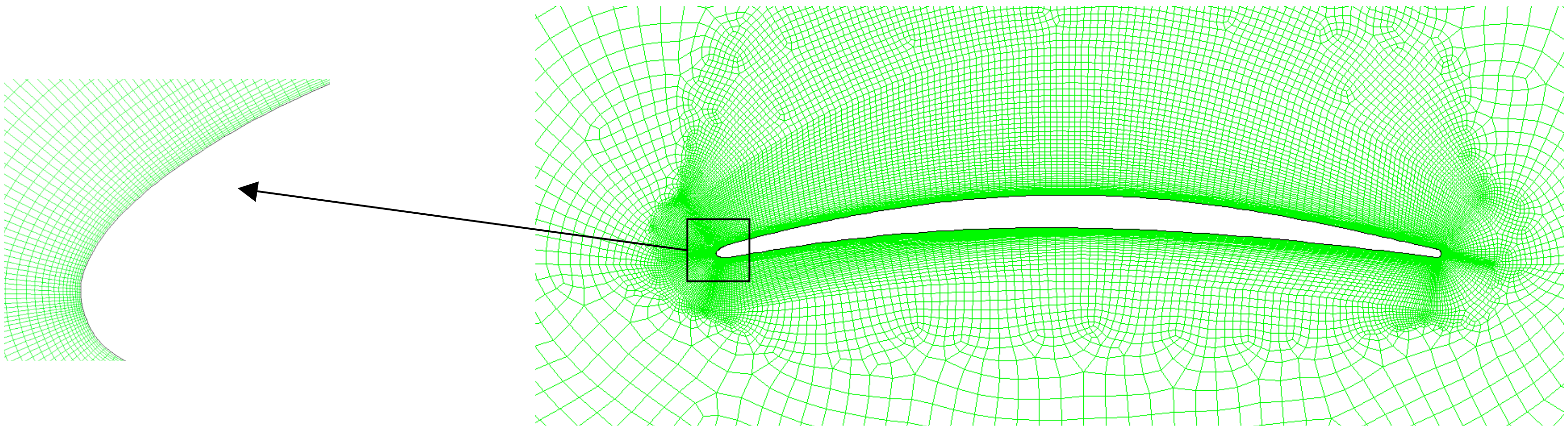




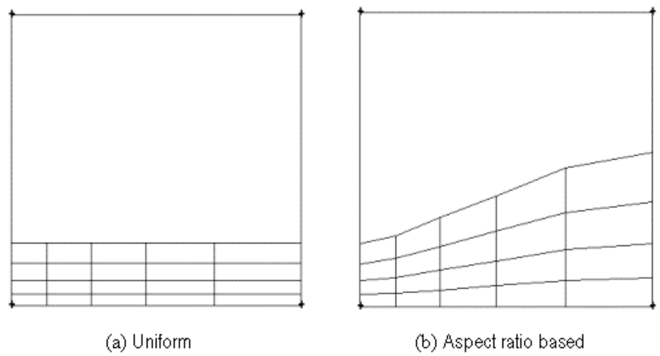
Grid generation – Clustering points

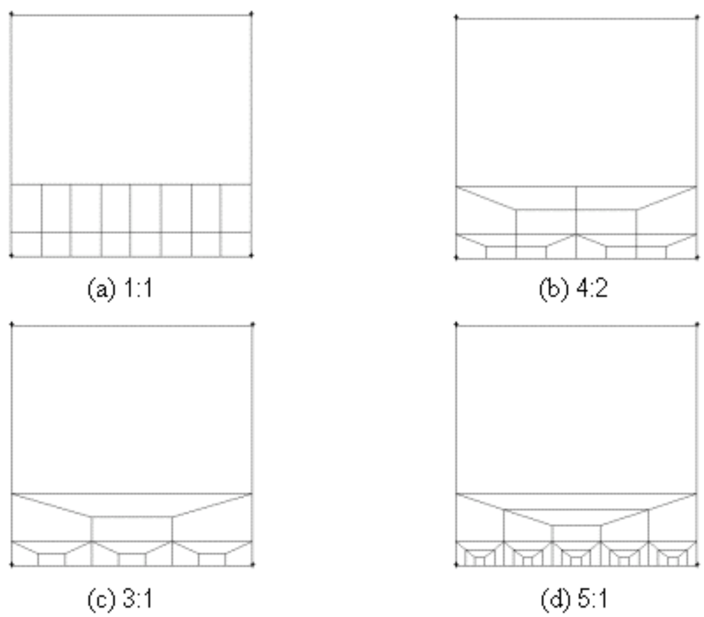
Sizing functions can be used effectively to define the size of the cells BUT they cannot provide directional control (anisotropy)

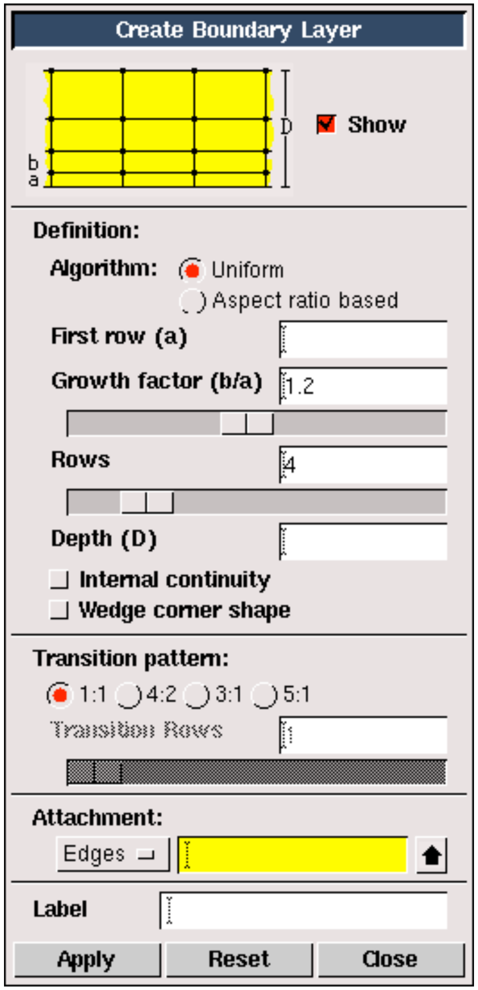
One option is to build (grow) elements from the boundaries and to form "viscous" layers



Grid generation – Boundary Layers

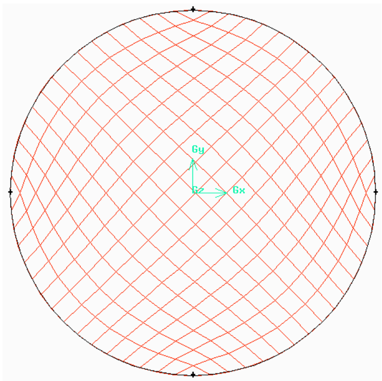




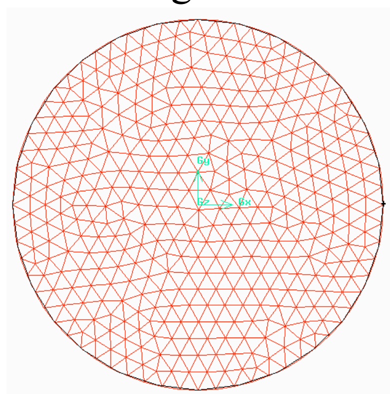


Example – meshing a circle

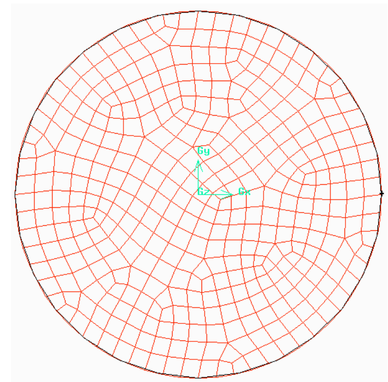
Mapping



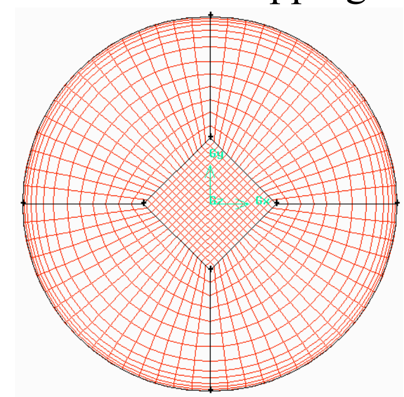
Triangulation



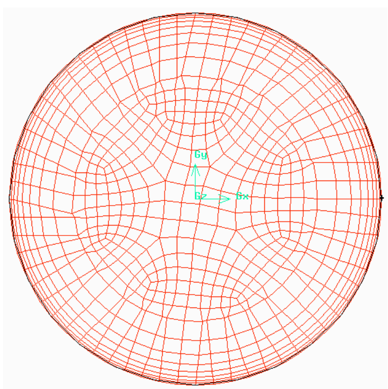
Paving



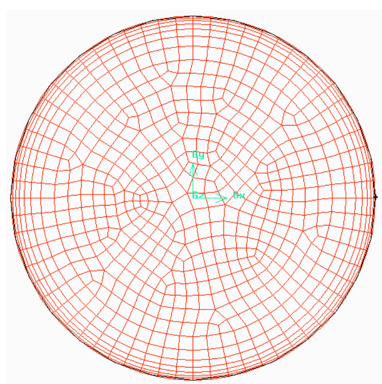
Multiblock mapping



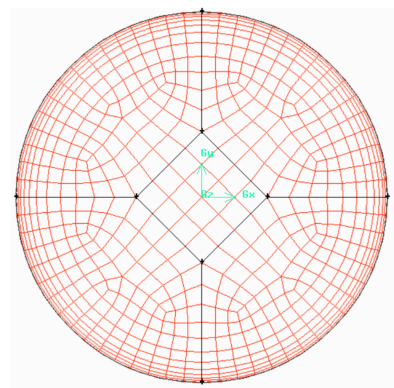
Boundary Layer
Paving



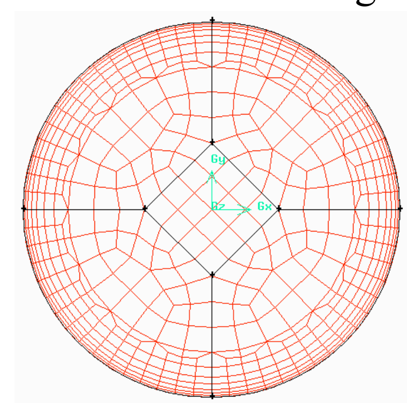
Boundary Layer
Paving



Boundary Layer
Multiblock Paving



Boundary Layer Transition
Multiblock Paving



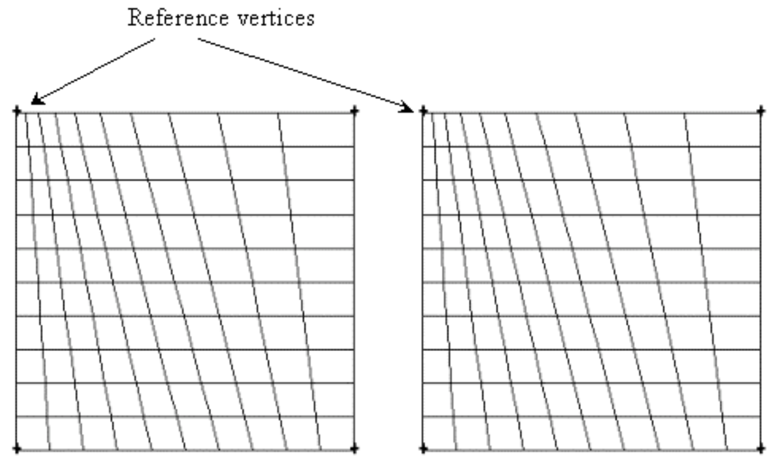
Circle defined as segments

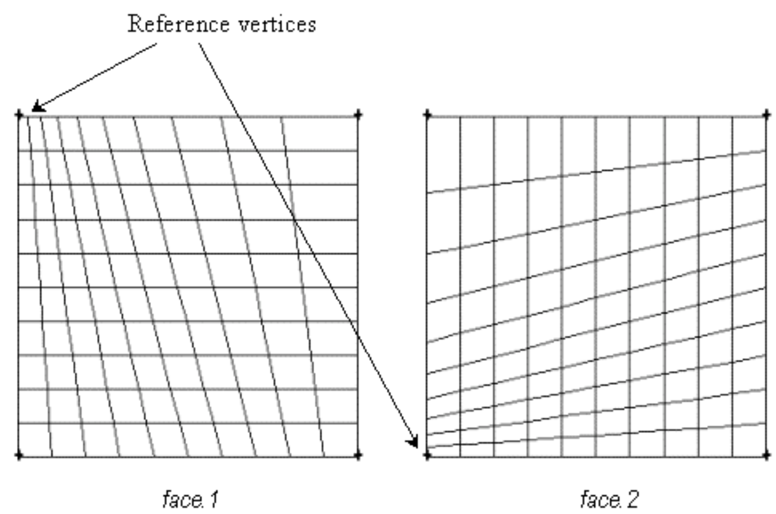
Mesh linking

Edges, faces and volume meshes can be linked

Define corresponding entities and ALSO reference entities

Needed to “enforce” coincident grids on different entities (i.e. for periodicity bc)





Grid quality

Quality measures are NOT absolute but should be considered in connection with solution schemes

The final accuracy of a procedure is ALWAYS a function of the grid quality

Several geometrical measures can be defined:

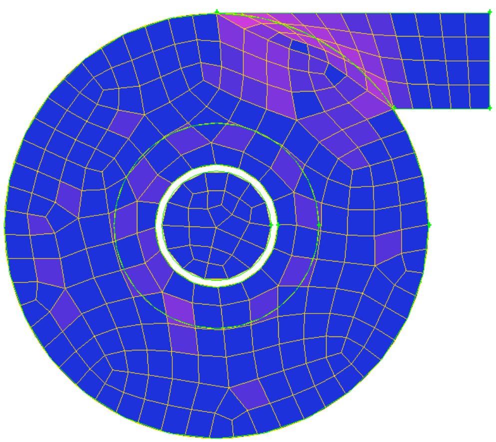
- Depending on the size of the elements
- Depending on the shape of the elements
- Depending on relative dimensions of neighboring elements

Quality measures available in GAMBIT

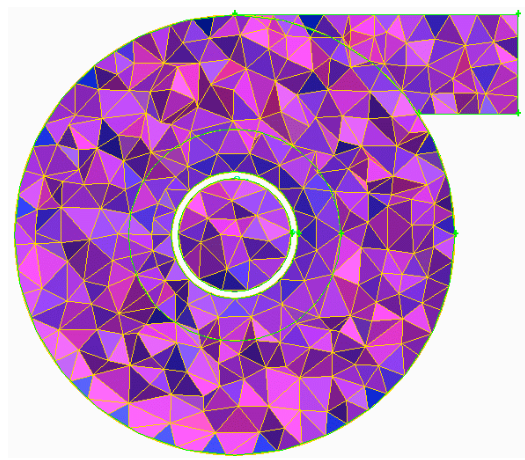
Quality Type	*2-D Element*		*3-D Element*			
						
Area	X	X				
Aspect Ratio	X	X	X	X	X	X
Diagonal Ratio	X		X			
Edge Ratio	X	X	X	X	X	X
EquiAngle Skew	X	X	X	X	X	X
EquiSize Skew		X		X		
MidAngle Skew	X		X			
Stretch	X		X			
Taper	X		X			
Volume			X	X	X	X
Warpage	X					

Examine meshes

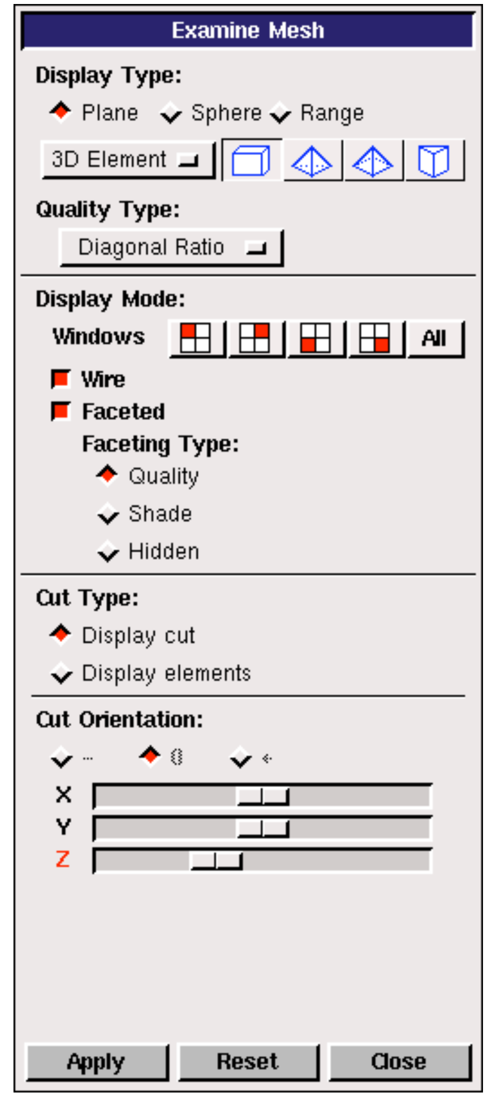
- Define mesh element to examine
- Define a cutting plane
- Define the quality measure



Aspect ratio

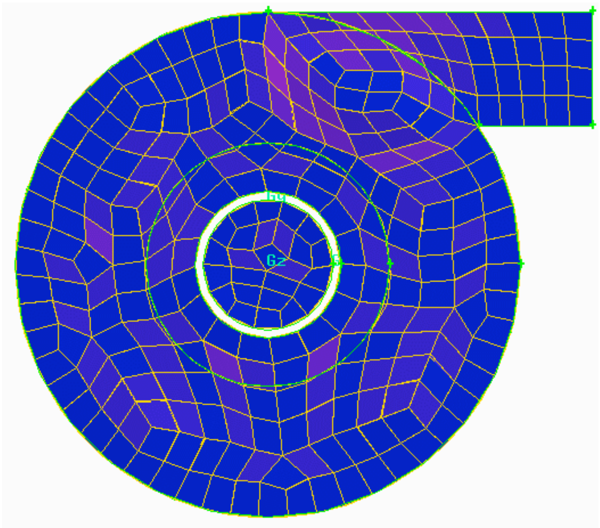


Equiangular skew

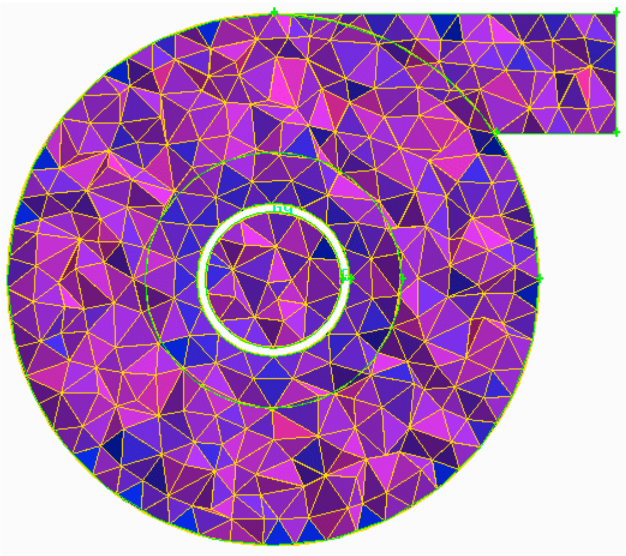


Mesh improvement

• Smoothing operators are applied to redistribute nodes



Aspect ratio

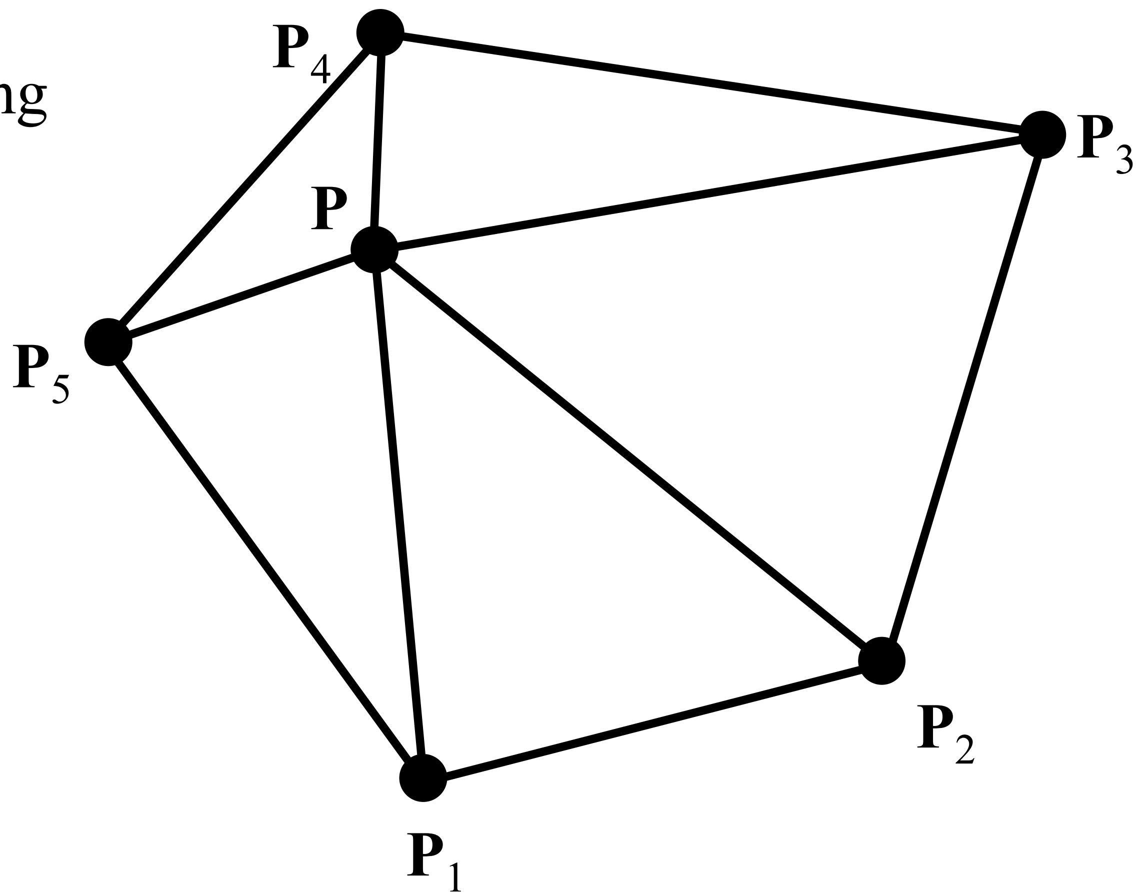


Equiangular skew

Typically, improvement in 3D meshes are based on improved 2D meshes

Mesh improvement/smoothing

Averaging
Methods

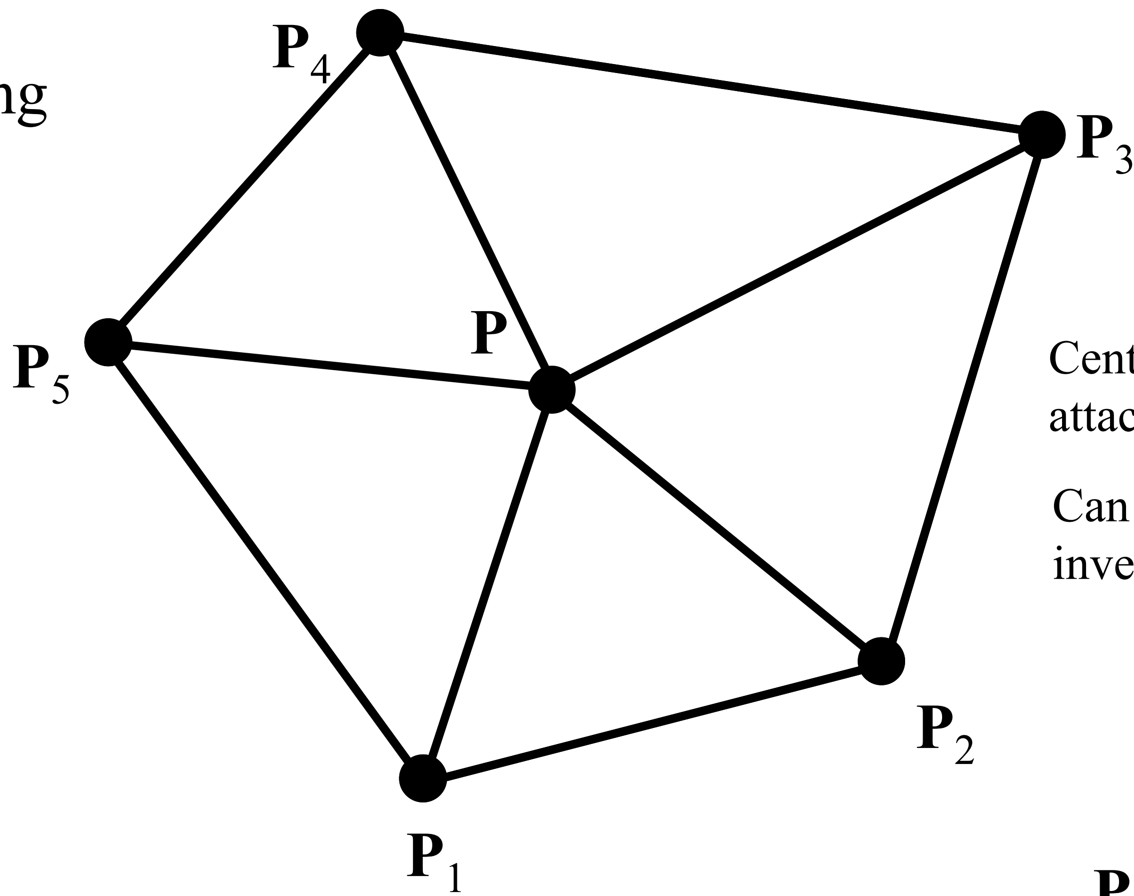


Laplacian

(Field, 1988)

Mesh improvement/smoothing

Averaging Methods



Centroid of attached nodes

Can create inverted elements

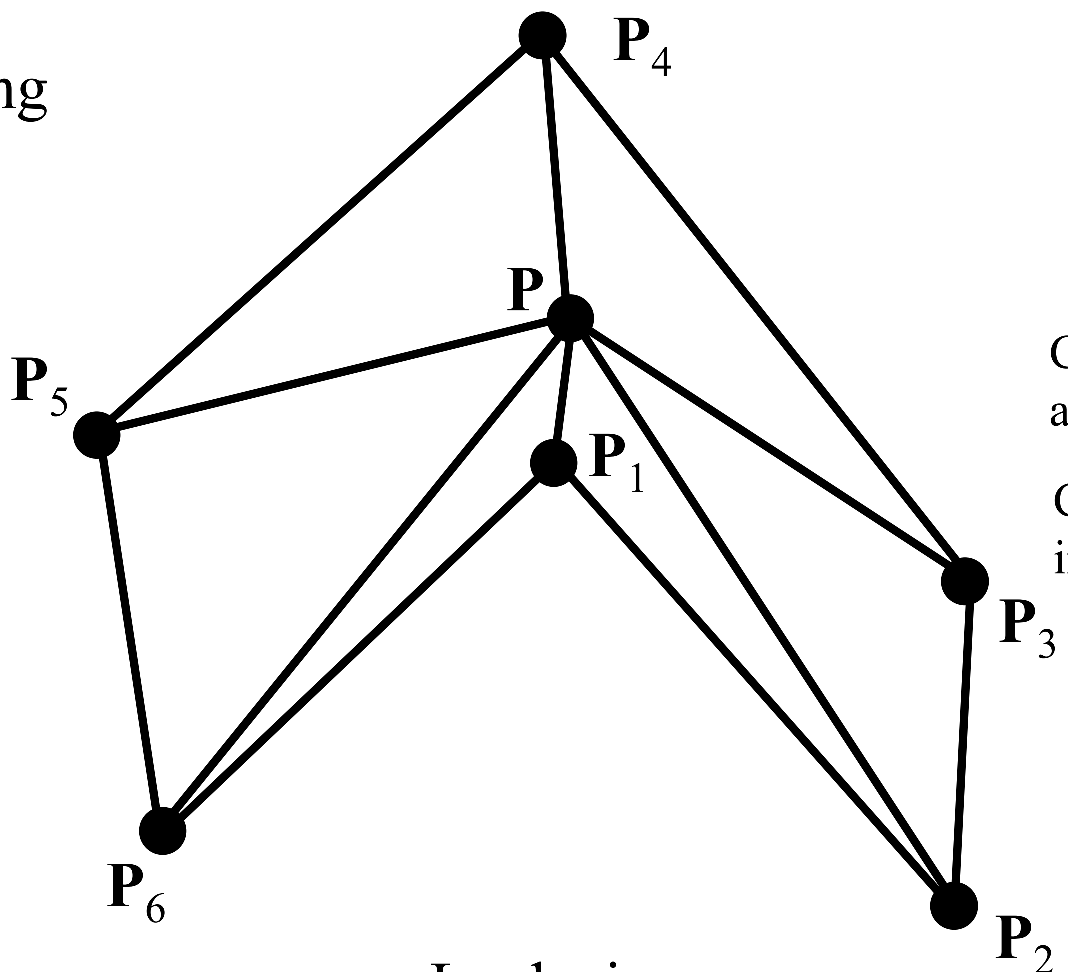
$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

(Field, 1988)

Laplacian

Mesh improvement/smoothing

Averaging Methods



Centroid of attached nodes

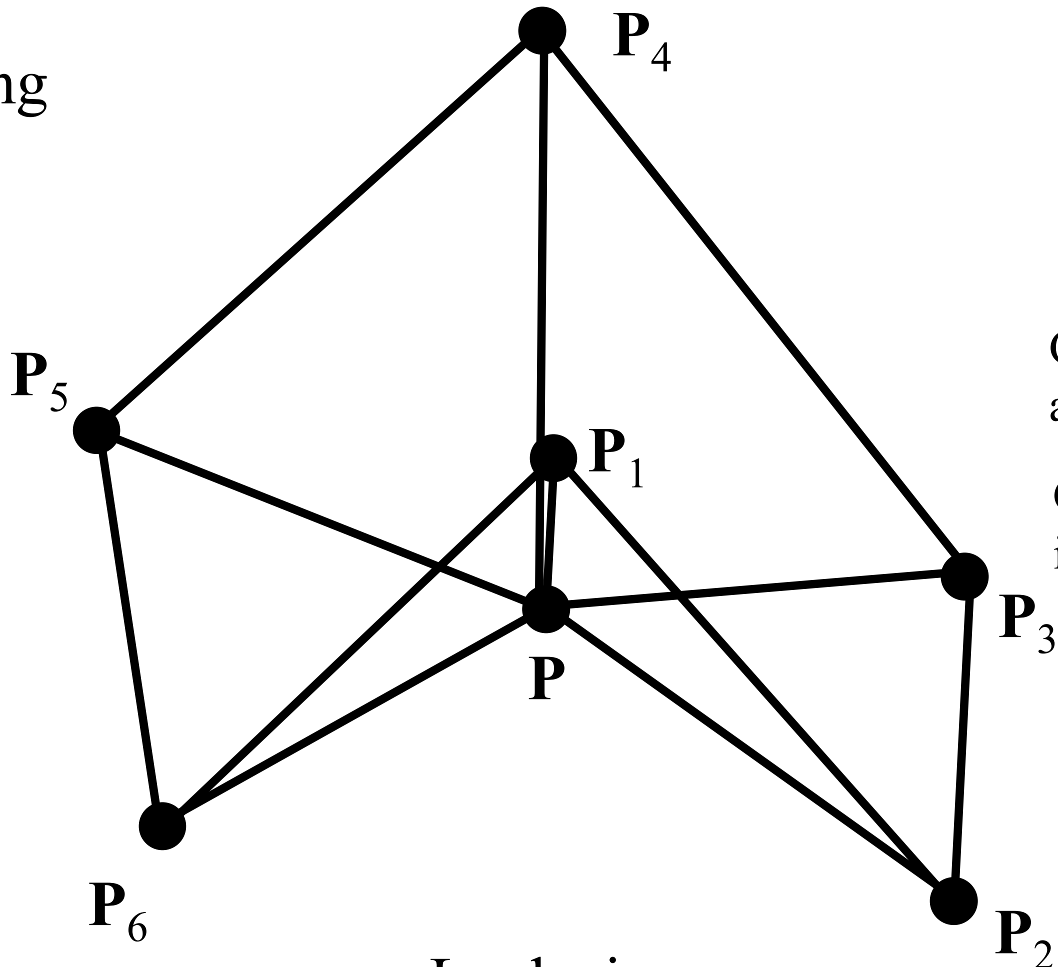
Can create inverted elements

$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

Laplacian

Mesh improvement/smoothing

Averaging Methods



Centroid of attached nodes

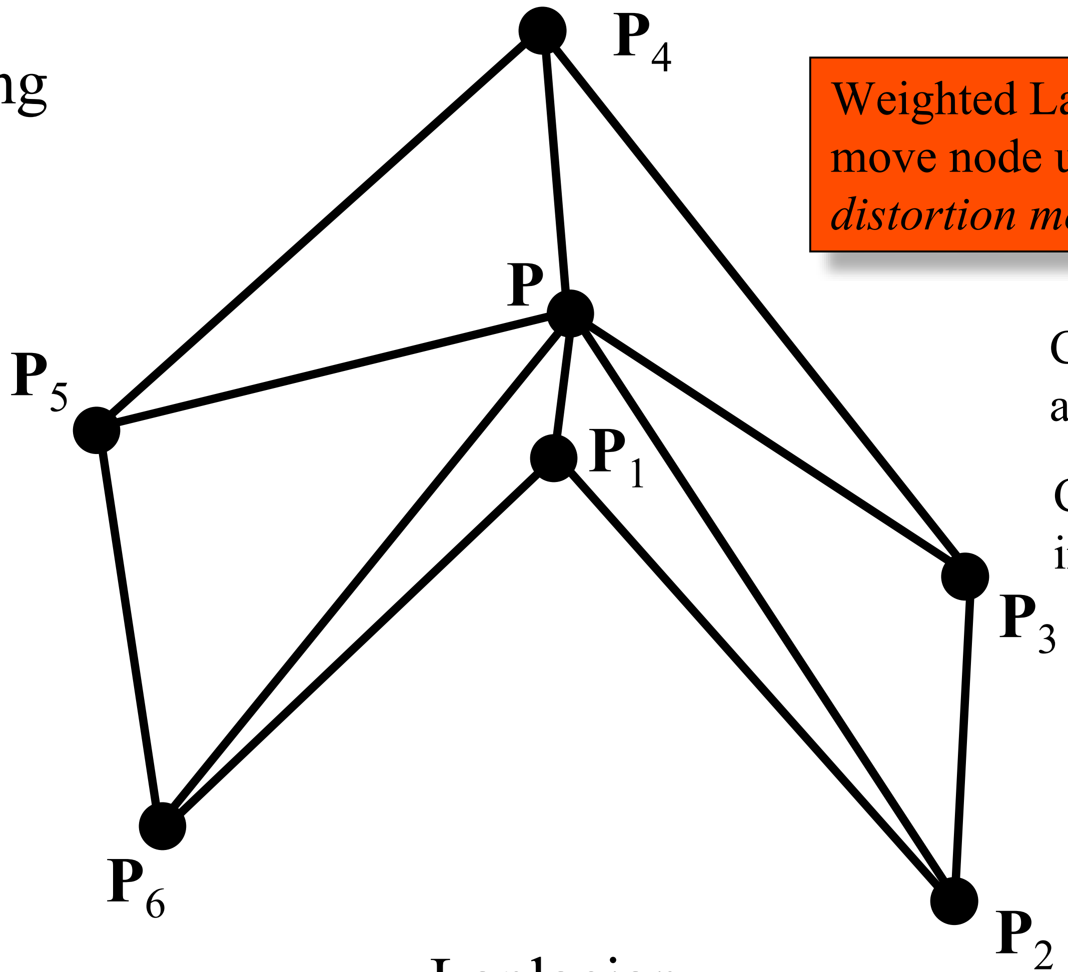
Can create inverted elements

$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

Laplacian

Mesh improvement/smoothing

Averaging Methods



Weighted Laplacian: Do not move node unless minimum *distortion metric* is improved

Centroid of attached nodes

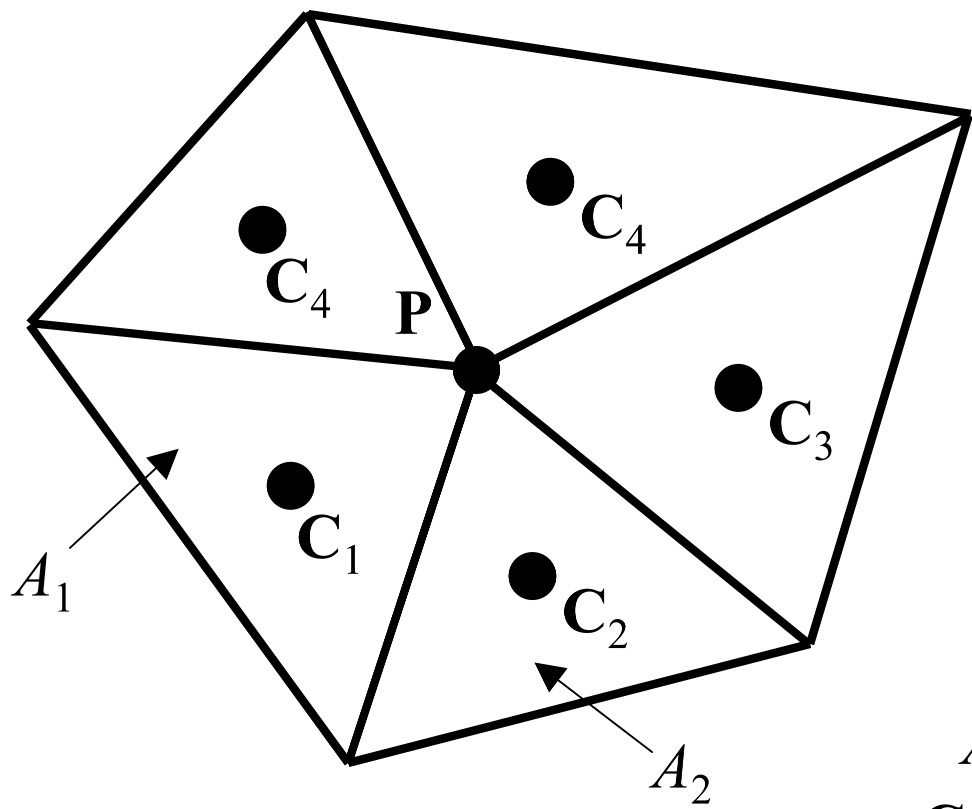
Can create inverted elements

$$\mathbf{P} = \frac{\sum_{i=1}^{n} \mathbf{P}_i}{n}$$

Laplacian

Mesh improvement/smoothing

Averaging Methods



Weighted average of triangle centroids

$$\mathbf{P} = \frac{\sum_{i=1}^{n} A_i \mathbf{C}_i}{\sum_{i=1}^{n} A_i}$$

A_i=area of triangle i

$\mathbf{C}_i$=centroid of triangle i

Area Centroid Weighted

Grid Generation Automation

- GAMBIT saves a "journal" file with the commands issues during a session
- Journal file are ASCII editable files
- Commands are quasi-English and easy-to-use
- They are useful to trace-back sessions to find errors
- They can be made general by introducing User Defined Parameters

GAMBIT Journal file

The command:

```
Volume create width 1 depth 1 height 1 offset 0 0 0 brick
```

Generates a cube of size 1 centered at 0 0 0

On the other hand the sequence

```
$W = 2.3
$D = 1.5
$H = 4
Volume create width $W depth $D height $H offset 0 0 0 brick
```

Generates a cuboid of User-Specified Size centered at 0 0 0

GAMBIT Journal file

It is possible to perform operations on input parameters

```
$SUM = $A + 0.5*$B
```

Math. functions are available:

```
$SUM = SIN($A)
```

In addition, geometrical operations

```
$X = INTERSECTING(volume, “volume.5”, “volume.12”)
$X = BBOX("volume.3")
$X = GETNORMAL("face.3", 1, 13, 89)
(…)
```

GAMBIT Journal file

Conditional statements:

```
if cond ($A .eq. 5)
   volume create sphere radius ($A+3)
endif
```

Loops:

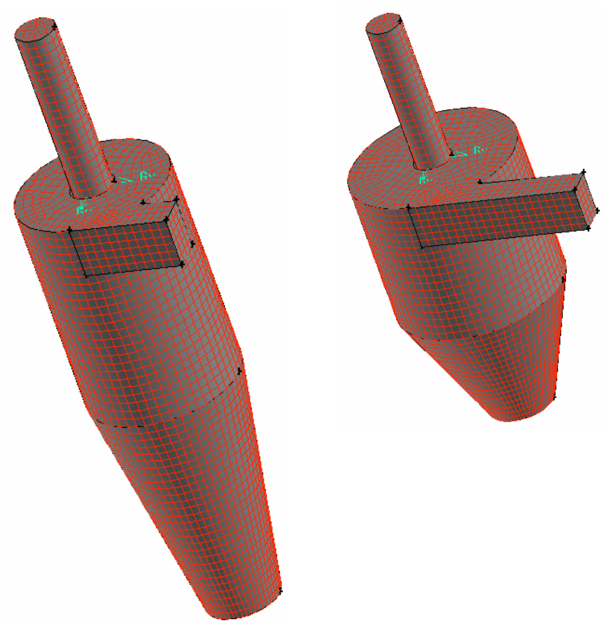
```
$Z = 0
do para "$Z" init 6 cond ($Z .le. 24) incr ($Tmp*3)
       volume create sphere radius $Z
Enddo
```

Relation and logical operators :

```
.eq. .le. .lt. (…)

.and. .or.
```

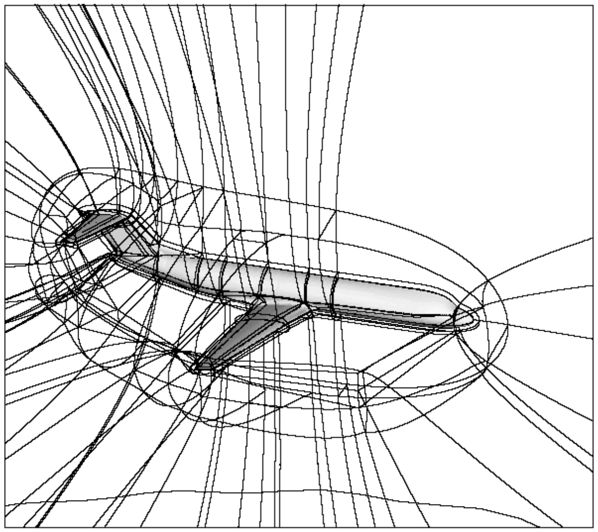
Example of Journal file



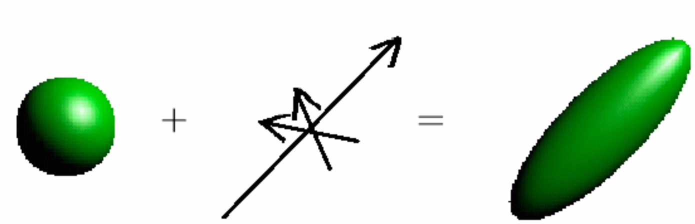
```
/ --------------------------------------------------------
/ CYCLONE GRID GENERATION
/ ME269B - Spring 2002
/ --------------------------------------------------------
/
/ R1 = External radius of Cyclone
/ R2 = Gas Outlet Pipe (External)
/ R3 = Gas Outlet Pipe (Internal)
/ RB = Particles Outlet (Bottom)
/ H1 = Height of the Cylindrical Part of Cyclone
/ H2 = Height of the Conical Part
/ HE = Depth of the Outlet Channel into the Cyclone
/
/ inletl = Length (x) of the Gas Inlet Channel
/ inleta = Height (z) of the Gas Inlet Channel
/ inletb = Span (y) of the Gas Inlet Channel
/ outletl = Length (z) of the outlet (gas) pipe
/
/ cellsize = Average size of the cells
/
/ Remark: z-axis is the Cyclone Axis
/ --------------------------------------------------------
/
/ Input Quantities
/
$R1 = 1.555
$R2 = 0.45
$R3 = 0.4
$RB = 0.75
$H1 = 4.5
$H2 = 5
$HE = 3.6
$inletl = 2
$inleta = 1.03
$inletb = 0.7
$outletl = 7.2
$cellsize = 0.18
/
/
```

Grid generation research

- Structured grids: automatic generation of mappable subdomains
  NLR 2000-366 Report
  (PDF available from class web site)

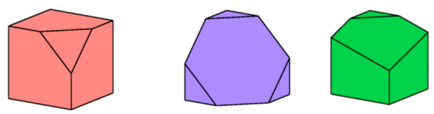


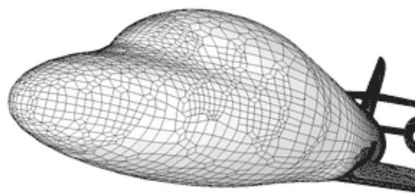
- Unstructured tetrahedral grids: anisotropic Delaunay schemes
  Shimada et al. "High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing" (PDF available)



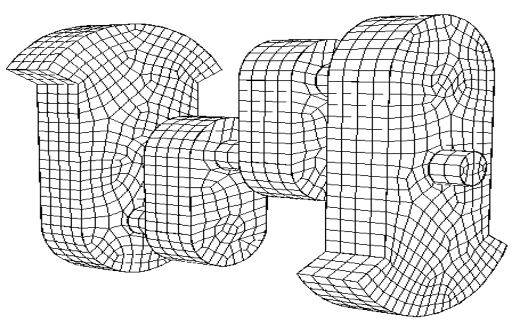
Grid generation research

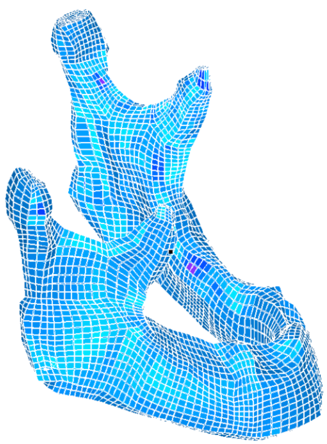
- Unstructured hex-dominant grids: OCTREE based
  SAMM – Computational Dynamics Ltd.
  Hexpress – Numeca International Inc.



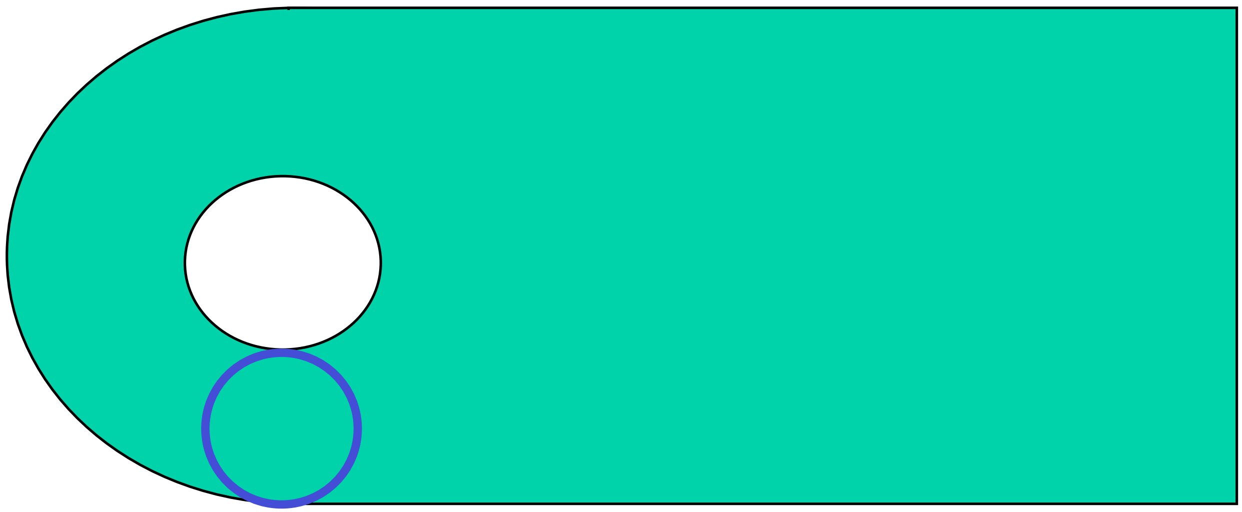


- Unstructured purely hexahedral grids: Whisker-Weaving
  CUBIT – Sandia National Lab. (PS report available)





Grid generation using Medial Axis

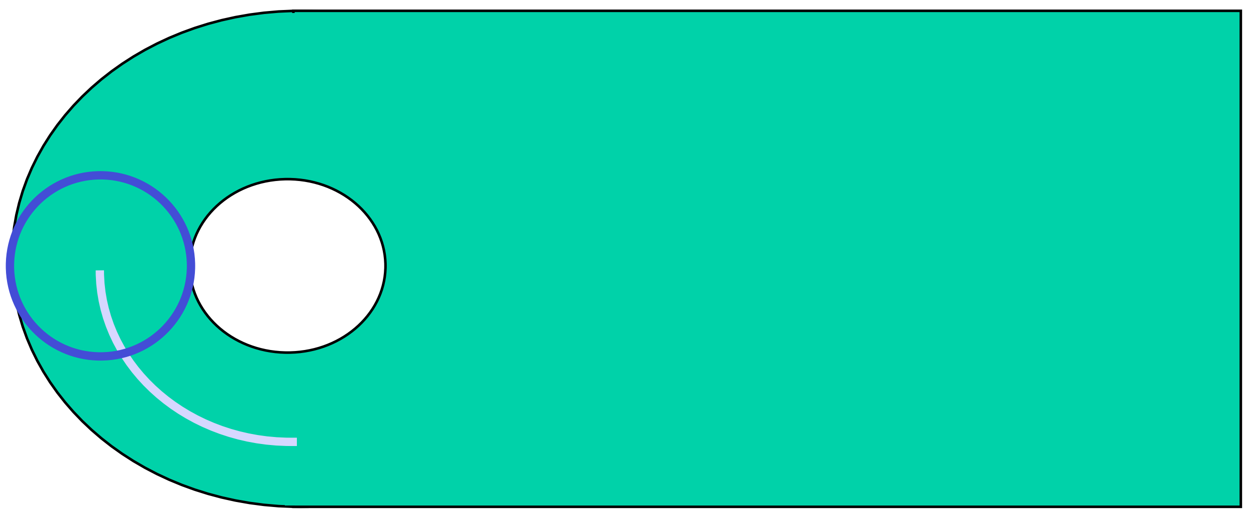


Medial Axis

•Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
•Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Grid generation using Medial Axis



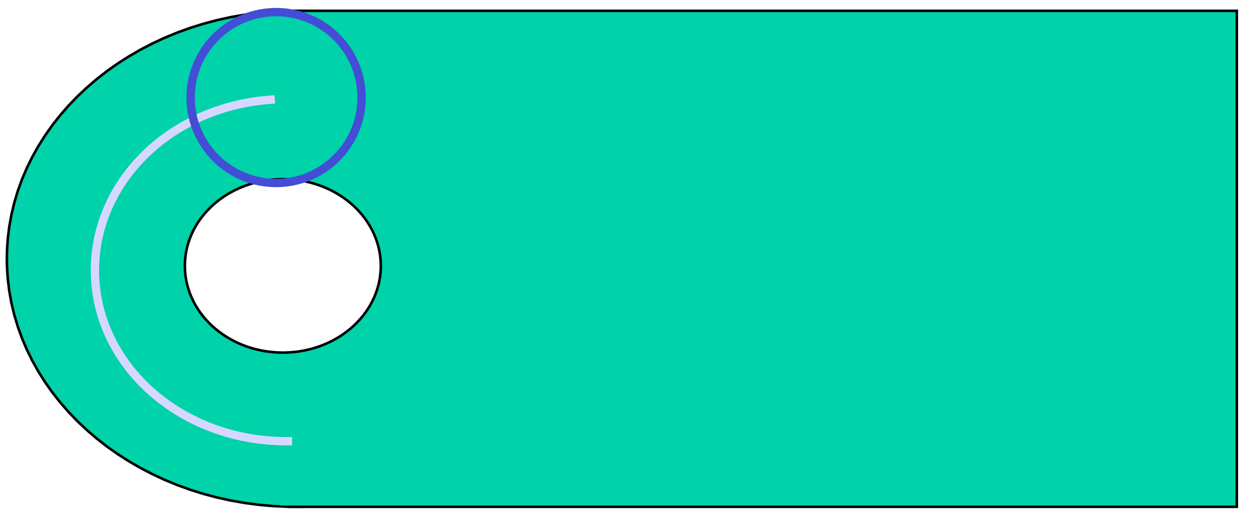
Medial Axis

•Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object

•Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Grid generation using Medial Axis



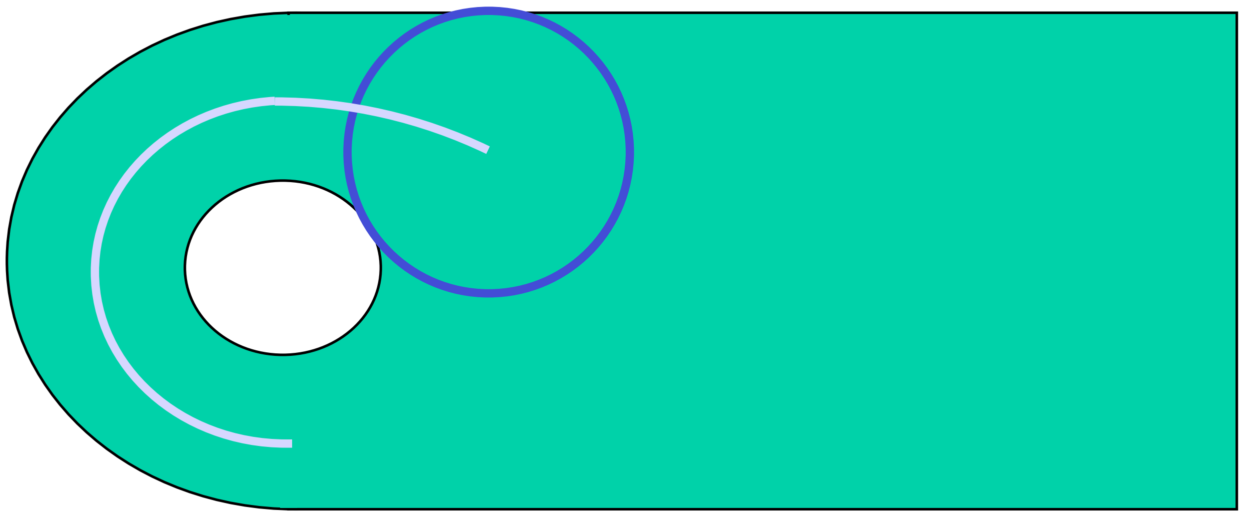
Medial Axis

•Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object

•Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Grid generation using Medial Axis

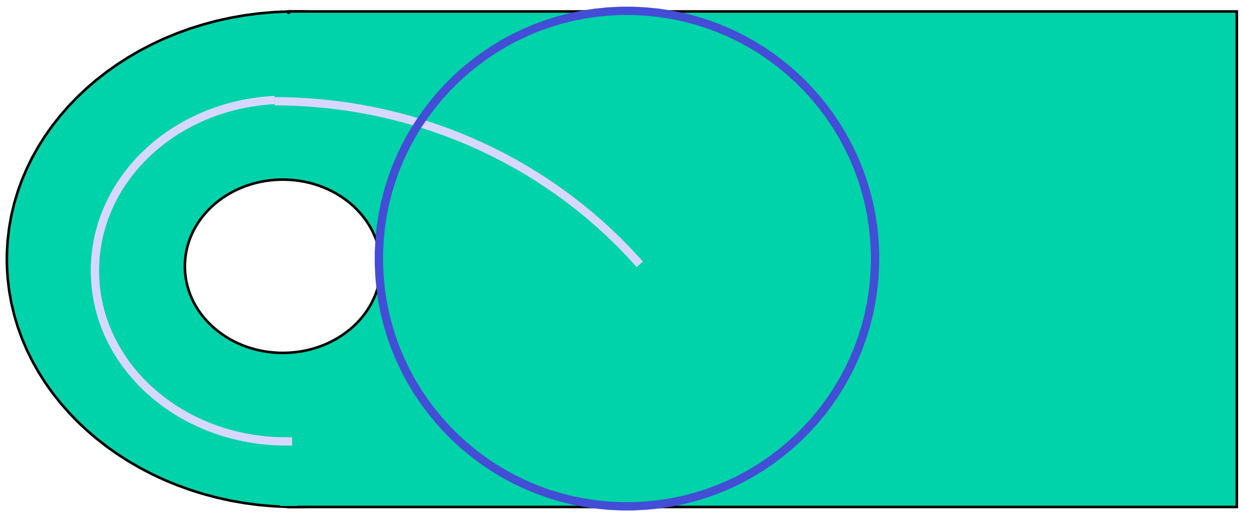


Medial Axis

•Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object
•Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Grid generation using Medial Axis



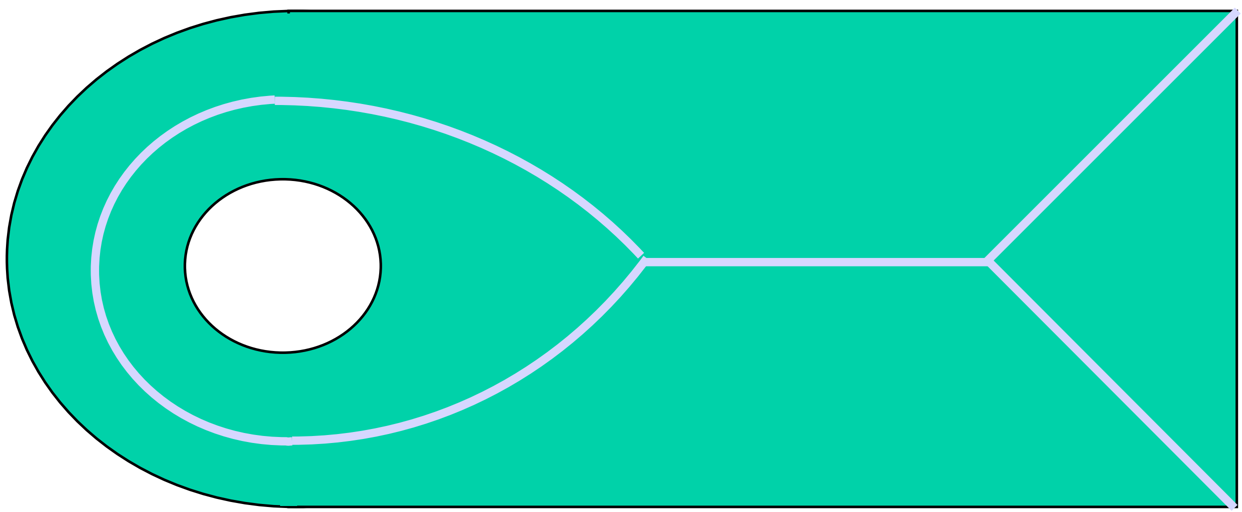
Medial Axis

•Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object

•Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Grid generation using Medial Axis



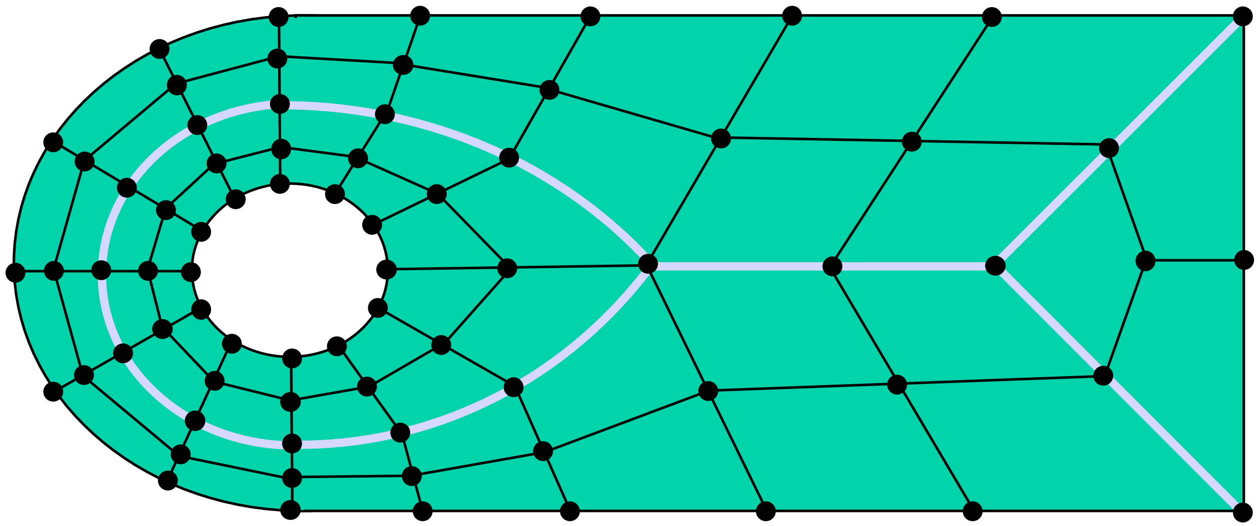
Medial Axis

•Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object

•Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Grid generation using Medial Axis



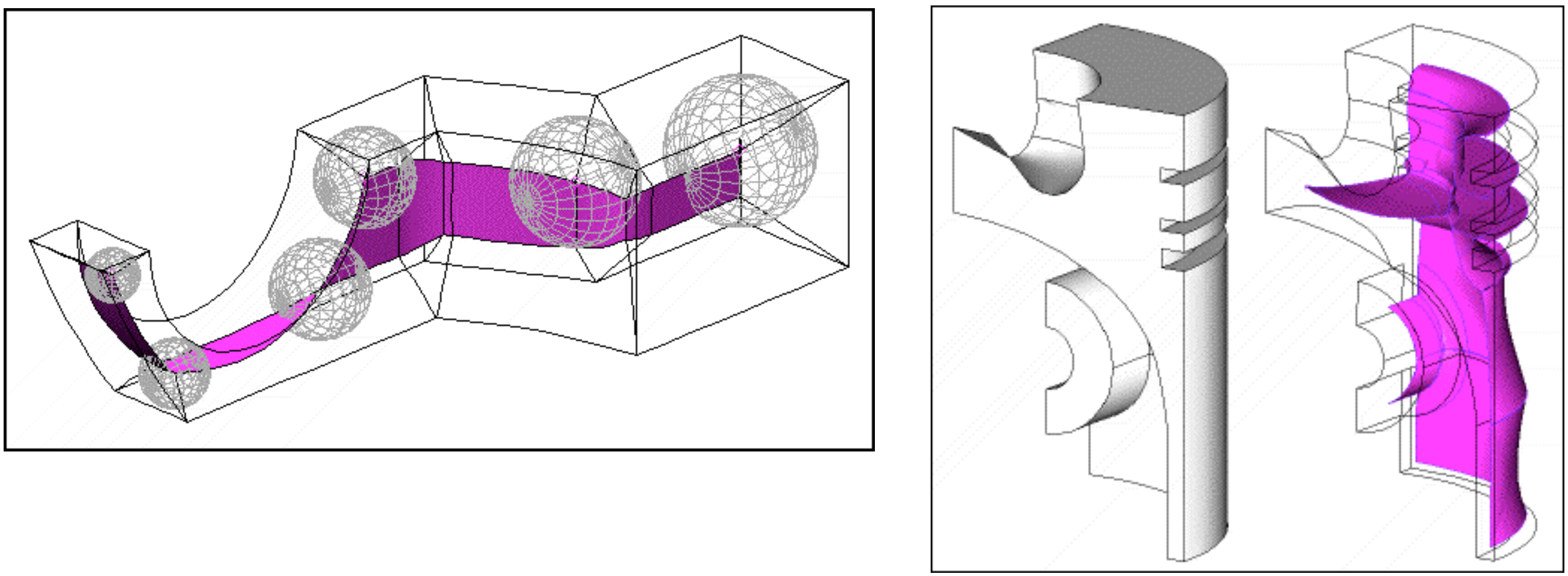
Medial Axis

•Medial Object - Roll a Maximal circle or sphere through the model. The center traces the medial object

•Medial Object used as a tool to automatically decompose model into simpler mapable or sweepable parts

(Price, 95;97)(Tam,91)

Grid generation using Medial Axis



3D Medial Object examples
(from FEGS website, URL: http://www.fegs.co.uk/medial.html)

Grid generation – Links and References

- *Links*
  - *Mesh generation and grid generation on the Web*
    http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html
  - *Meshing research corner*
    http://www.andrew.cmu.edu/user/sowen/mesh.html
  - *General CFD: Topic mesh generation*
    http://www.cfd-online.com

- *References:*
  - Handbook of grid generation. Thompson, Soni, Weatherill, CRC Press
  - Numerical Grid Generation: Foundation & Applications.
    Thompson, Warsi, Mastin. North Holland Press